College of
William & Mary
Department of Computer Science

WM-CS-2005-03

Nearly optimal preconditioned methods for hermitian eigenproblems under limited memory. Part I: Seeking one eigenvalue

Andreas Stathopoulos

July 2005

NEARLY OPTIMAL PRECONDITIONED METHODS FOR HERMITIAN EIGENPROBLEMS UNDER LIMITED MEMORY. PART I: SEEKING ONE EIGENVALUE *

ANDREAS STATHOPOULOS †

Abstract. Large, sparse, Hermitian eigenvalue problems are still some of the most computationally challenging tasks. Despite the need for a robust, nearly optimal preconditioned iterative method that can operate under severe memory limitations, no such method has surfaced as a clear winner. In this research we approach the eigenproblem from the nonlinear perspective that helps us develop two nearly optimal methods. The first extends the recent Jacobi-Davidson-Conjugate-Gradient (JDCG) method to JDQMR, improving robustness and efficiency. The second method, Generalized-Davidson+1 (GD+1), utilizes the locally optimal Conjugate Gradient recurrence as a restarting technique to achieve almost optimal convergence. We describe both methods within a unifying framework, and provide theoretical justification for their near optimality. A choice between the most efficient of the two can be made at runtime. Our extensive experiments confirm the robustness, the near optimality, and the efficiency of our multimethod over other state-of-the-art methods.

1. Introduction. The numerical solution of large, sparse, Hermitian and real symmetric eigenvalue problems is central to many applications in science and engineering. It is also one of the most time consuming tasks. Recently, electronic structure calculations, with eigenproblems at their core, have displaced Quantum Chromodynamics as the top supercomputer cycle user. The symmetric eigenvalue problem seems deceptively simple to solve, given well conditioned eigenvalues and a wealth of theoretical knowledge. Yet, these advantages have enabled researchers to push modeling accuracy to unprecedented levels, routinely solving for a few extreme eigenvalues of matrices of size more than a million, while an order of a billion has also been tried [48].

The sheer size of these problems can only be addressed by iterative methods. At the same time, the size imposes limits on the memory available to these methods. Moreover, preconditioning becomes imperative to reduce the total number of iterations. While many eigenvalue iterative methods have been proposed, there is little consensus on which method is the best and in what situations. The unrestarted Lanczos method is known to be optimal for solving Hermitian eigenvalue problems but, unlike the Conjugate Gradient (CG) method for linear systems, it requires unlimited storage of its iteration vectors. With preconditioning, or under limited storage, the question of optimality remains open. Furthermore, there is a noticeable scarcity of high quality, general purpose software for preconditioned eigensolvers. In this research we seek an optimal, or nearly optimal, method that can utilize preconditioning and that can be implemented in a robust software that is also flexible.

In the particular case of seeking one eigenpair, if the eigenvalue were known, solving a linear system using CG to obtain the corresponding eigenvector would yield the optimal Lanczos convergence. In practice both the eigenvalue and the eigenvector are unknown, so the appropriate way to approach this problem is through the resulting slight nonlinearity, which then helps us to identify practical and near optimal methods. Currently, the best methods follow this approach, adopting either the Newton point of view or the nonlinear Conjugate Gradients (NLCG) point of view.

In the first part of this paper we argue that regardless of the approach (Newton or NLCG), the underlying iterative method should be the Generalized Davidson method. In the second part, we examine the optimality from the Newton point of view, and identify a truncated Newton version followed by JDCG [46] as within a small factor of optimal. In fact, we argue

*Work supported by National Science Foundation grants: ITR/DMR 0325218, ITR/AP-0112727.

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

that this factor is less than three and usually much less than two. We then extend JDCG to JDQMR by adapting the quasi-minimal residual (QMR) instead of CG, that allows for better stopping criteria of the inner iteration, for indefinite preconditioners, interior eigenvalues and improved robustness. In the third part of the paper, we study the NLCG point of view and, in particular, the related locally optimal CG recurrence. The approach can be used for short recurrences (as in LOBPCG [32]) or for restarting the subspace accelerated GD+1 [44, 68]. We show why local optimality of the Ritz value extends also to the Ritz vector, and we provide new intuition on the performance of GD+1 by relating it to the limited memory quasi-Newton methods (such as L-BFGS) [45]. In the fourth part, we present one of the most extensive set of experiments in the literature, comparing our methods with state-of-the-art methods.

Our conclusion to be drawn from our experiments is that both GD+1 and JDQMR always solve a given problem, regardless of the difficulty, and one of the two is always the fastest method. Over all the methods tested, GD+1 typically yields the smallest number of iterations, often matching the optimal convergence. The cheaper inner iteration of JDQMR benefits cases where the preconditioner and the matrix-vector operations are inexpensive. As this expense can be measured at runtime, an implementation based on the common GD driver can switch dynamically between GD+1 and JDQMR to make the most effective use of these two nearly optimal methods.

We note that this is not a review paper, although substantial amount of review material is used to identify, build, and provide new intuition on the various components of the desired method. To keep the paper tractable both in size and readability, we focus on finding only one eigenpair. The additional problems of deflation, locking, correction equation projectors, blocking, and subspace acceleration which are relevant when a large number of eigenvalues are required will be addressed in a separate paper.

2. The Generalized-Davidson as an outer iteration model. There is a plethora of iterative methods for large, sparse, symmetric eigenvalue problems. Invariably, these are Krylov or Krylov-like subspace methods that approximate the required eigenvectors from a subspace that usually grows in size as it is iteratively updated. The methods host a variety of differentiating features, of which two are the defining features for our research: We need methods that can utilize arbitrary preconditioners and that can operate on a limited memory space.

Shift-and-invert is probably the most well studied and powerful preconditioning approach [52], which when coupled with a Lanczos-type method can produce robust, industrial strength methods [27]. In this research we assume that an exact factorization for shift-and-invert is not possible because of the size of the problem. In such cases, approximations to $(A-\eta I)^{-1}$ are often used as preconditioners to linear solvers for the Inverse Iteration (INVIT), Rayleigh Quotient Iteration (RQI) or the (Jacobi-) Davidson methods [52, 71, 43, 60]. Although the accuracy of the term "eigenvalue preconditioning" for this approach has received some scrutiny [15, 31], the resulting methods improve convergence similarly to the corresponding preconditioned linear system solvers. Moreover, there are many established and tunable ways to produce such preconditioners [56]. Some applications may use different types of preconditioners, including polynomial transformations, frequency filtering or smoothing, or other techniques that do not correspond to approximately solving a linear system [55, 67].

The Generalized Davidson (GD) method was one of the first methods to cater to such an arbitrary selection of preconditioners [43]. Extending the original Davidson's method [11], at every iteration GD applies a user defined function on the residual of the sought eigenpair to "precondition" it — or to enrich it in the required directions. The vector is then used to enlarge the search space from which a new approximation is obtained through the Rayleigh-Ritz

ALGORITHM 2.1. *The Generalized Davidson algorithm for one eigenpair*

start with $\mathbf{v}_0$ *starting vector*
$\mathbf{t}^{(0)} = \mathbf{v}_0,\ m = 0,\ nmv = 0$
while $nmv < max\_num\_matvecs$
    Orthonormalize $\mathbf{t}^{(m)}$ *against* $\mathbf{v}_i, i = 1, \ldots, m$
    $m = m + 1$; $nmv = nmv + 1$; $\mathbf{v}_m = \mathbf{t}^{(m-1)}$; $\mathbf{w}_m = A\mathbf{v}_m$
    $H_{i,m} = \mathbf{v}_i^T \mathbf{w}_m$ *for* $i = 1, \ldots, m$
    compute eigendecomposition $H = S\Theta S^T$ *with* $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_m$
    $\mathbf{u}^{(m)} = Vs_1$; $\theta^{(m)} = \theta_1$; $\mathbf{w}^{(m)} = Ws_1$
    $\mathbf{r}^{(m)} = \mathbf{w}^{(m)} - \theta^{(m)}\mathbf{u}^{(m)}$
    if $\|\mathbf{r}^{(m)}\| \leq tol$, **return** $\theta^{(m)}, \mathbf{u}^{(m)}$
    if $m \geq m_{max}$ *then*
        $H = 0$
        for $i = 2, \ldots, m_{min}$
            $\mathbf{v}_i = Vs_i$; $\mathbf{w}_i = Ws_i$; $H_{ii} = \theta_i$
        end for
        $\mathbf{v}_1 = \mathbf{u}^{(m)}$; $\mathbf{w}_1 = \mathbf{w}^{(m)}$; $H_{11} = \theta^{(m)}$; $m = m_{min}$
    end if
    Precondition the residual $\mathbf{t}^{(m)} = Prec(\mathbf{r}^{(m)})$
end while

procedure [52]. When the search space exhausts the available memory, the method restarts with the best approximations of the required eigenvectors and possibly with additional information to improve convergence. Algorithm 2.1 depicts a version of the basic GD algorithm for finding one, smallest eigenpair $(\lambda_1, \mathbf{x}_1)$ of a real symmetric matrix, A.

Despite many efforts over two decades, the theory behind the convergence of the GD algorithm is still not well understood [43, 10, 69, 60]. For some recent, promising results see [49, 50, 34, 47]. One of the early unsettling issues with the GD had been the stagnation of the method if an accurate enough approximation to $(A - \theta^{(m)}I)^{-1}$, where $\theta^{(m)}$ is the Ritz value, is used as a preconditioner. An elegant resolution to this problem has been given by Sleijpen et al. with the Jacobi-Davidson (JD) method [60]. Instead of inverting $(A - \eta I)$, they showed that the appropriate form for this preconditioning should be an approximate solution to the correction equation:

$$(I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T})(A - \eta I)(I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T})\mathbf{t}^{(m)} = -\mathbf{r}^{(m)} = \theta^{(m)}\mathbf{u}^{(m)} - A\mathbf{u}^{(m)}, \tag{2.1}$$

where η is a shift close to the wanted eigenvalue. Thus, by working orthogonally to the Ritz vector $\mathbf{u}^{(m)}$, JD avoids stagnation. Among many known properties for this correction equation, one that is relevant to this paper is that if eq. (2.1) is solved accurately, the new vector $\mathbf{u}^{(m)} + \mathbf{t}^{(m)}$ is collinear to the iterate produced by Inverse Iteration [41, 60, 58, 15, 73]. In addition, if $\eta = \theta^{(m)}$, the method is equivalent to RQI with subspace acceleration of all previous iterates. The true flexibility of GD and JD, however, is that they converge even when eq. (2.1) is solved approximately through an iterative method such as CG. A preconditioner K can also be used as long as it is inverted orthogonally to the space of $Q = \mathbf{u}^{(m)}$. To this effect, the pseudoinverse of such a preconditioner can be written as:

$$\big((I - QQ^T)K(I - QQ^T)\big)^+ = (I - K^{-1}Q(Q^TK^{-1}Q)^{-1}Q^T)K^{-1}(I - QQ^T) \tag{2.2}$$

$$= K^{-1}(I - Q(Q^TK^{-1}Q)^{-1}Q^TK^{-1})(I - QQ^T). \tag{2.3}$$

Moreover, CG can be implemented efficiently with only one projection with Q per iteration. See [17, 60, 3] for details.

Algorithm 2.1 is identical to the JD algorithm as provided in [3], except for step (19) which is handled specially in JD. In this sense, we can view JD as a subcase of GD. Practitioners, however, usually refer to GD as a *single* application of the available preconditioner K^{-1} to the residual. In that case, preconditioning orthogonally to $\mathbf{u}^{(m)}$, using equations (2.2–2.3) is usually called Olsen's method [48] (or JD with no inner iterations), and it involves two operations with K^{-1} which can be very expensive. In practice, K does not often approximate $(A-\theta^{(m)}I)$, but A, and even when it does it is rarely accurate enough to require or even benefit from the JD projections, and thus GD is widely used in many applications [73, 2]. Exceptions exist for "non-standard" cases and when there is an inner iterative method as discussed in [61] and more extensively in [47]. In the remaining of this paper, we will use GD to refer to the method that preconditions the residual without Olsen's method.

Generality is an additional reason for considering GD as our basic iterative framework. Algorithm 2.1 is strikingly similar to an Arnoldi method with flexible preconditioner, making GD a chameleon method. Without preconditioning, GD is equivalent to Arnoldi, albeit with an expensive implementation, that facilitates not only the testing of software correctness, but also a common platform for comparison between methods, abstracting from the implementation details. For example, the restarted, unpreconditioned GD is mathematically equivalent to implicitly restarted Lanczos (IRL) and Thick Restarted Lanczos [63, 38, 42, 70, 74]. A block GD implementation is also possible [39, 65, 23], yielding the equivalents of block Lanczos [52] and subspace iteration [9] (without preconditioning), or preconditioned subspace iteration [3]. In addition, combining a block GD with our earlier GD+k restarting scheme [68], which we study closer in Section 4.3, allows an equivalent and stable implementation of the LOBPCG method [32]. Needless to say, with an appropriate solution to the correction equation, the algorithm is identical to many variants of RQI, INVIT, JD and the more recent JDCG of Notay that we study closely in the following section [46]. Beyond emulating the above methods, GD can enhance them with block, subspace acceleration, while working under given memory constraints.

Often the choice of algorithm and its parameters depends on the particular problem solved. The above discussion supports the argument that if there is to be one general purpose method that allows for arbitrary preconditioning and converges near optimally for a given problem, that algorithm must follow the GD template. The rest of the paper presents arguments why the tuning of the GD parameters, a dreaded task by practitioners, can be almost fully automated in the symmetric case.

3. The Newton view. Central to our discussion is the slight nonlinearity of the eigenvalue problem, i.e., both eigenvectors and eigenvalues are unknown. If the required eigenvalue λ_1 were known, the problem would degenerate to a symmetric linear system of equations $(A-\lambda_1 I)x_1 = 0$. This singular system can be solved optimally with CG or some other three-term recurrence method, provided that our initial guess x^0 is not defective in the direction of x_1, i.e., $x^0 = x_1 + y$ with $x_1 \perp y \in \mathrm{Range}(A-\lambda_1 I)$. A Krylov solver approximates the correction y from the subspace $\mathcal{K}((A-\lambda_1 I),(A-\lambda_1 I)y)$. Because $\mathcal{K}((A-\lambda_1 I),(A-\lambda_1 I)y) \subseteq \mathrm{Range}(A-\lambda_1 I)$, the solver will converge to x_1 with rate determined by the condition number of the deflated system: λ_{max}/λ_2. The optimality of CG implies that this is the ultimate convergence rate that any Krylov eigensolver can reach for finding the smallest eigenvalue [31, 33]. Note that unrestarted Lanczos achieves this rate, because the space that includes y yields also the eigenvalue λ_1. Equivalently, but more stably, we could solve eq. (2.1) for $t = y$ with $\eta = \lambda_1$, obtaining the same optimal rate and converging in one outer JD step. Because we use symmetric QMR, we call this benchmark QMRopt. In general, before reaching that rate, an eigensolver would also have to find λ_1.

This nonlinearity can be resolved by applying the Newton method on the Grassmann manifold (to enforce normalization of the eigenvectors), which is equivalent both to RQI and to the Jacobi-Davidson method with no subspace acceleration and with accurate solution of the correction equation [59, 73, 15, 57]. Convergence of the outer iterations is known to be ultimately cubic when the initial guess is sufficiently close to x_1 [52]. In the absence of a good initial guess, truncated Newton methods or a subspace method such as Lanczos or GD must be used to ensure global convergence.

It has long been noticed that using iterative methods to solve the linear equation in RQI and INVIT beyond some level of accuracy does not decrease the number of outer iterations, while increasing the overall number of matrix vector multiplications which typically correlates to the computational cost of the method. Interestingly, solving the linear equation approximately through a constant number of inner iterations has been shown to be equivalent to truncated (inexact) Newton methods [58]. However, the equivalence does not hold if the linear system in RQI is solved inexactly to a certain accuracy. Analysis of the stopping criteria for these inner-outer methods has been the focus of considerable research [53, 36, 21, 37, 62, 26, 25]. Because the solution vector for RQI and INVIT grows in norm the closer it is to the actual eigenvector, all proposed techniques test the convergence of the linear system relative to the norm of the solution vector. This norm can be monitored during the inner iteration.

The JD method is a better representative of inexact Newton minimization methods, because these apply the pseudoinverse of the Hessian to the gradient of the function. This is equivalent to solving eq. 2.1, through a certain number of inner iterations, or to a specified accuracy. Hence various stopping criteria for truncated Newton methods have been used in JD [60, 13]. Still, the objective is the convergence of the eigenvector, not of the linear system. Recently, an analysis for the interplay of inner-outer iterations was given by Notay [46]. Notay provided theoretical and experimental arguments why his method, JDCG, provides nearly optimal convergence.

3.1. The JDCG approach. To simplify the notation we denote the projected operators as:

$$A_{\eta,u^{(m)}} = (I-\mathbf{u}^{(m)}\mathbf{u}^{(m)T})(A-\eta I)(I-\mathbf{u}^{(m)}\mathbf{u}^{(m)T}) \tag{3.1}$$

$$K_{u^{(m)}} = (I-\mathbf{u}^{(m)}\mathbf{u}^{(m)T})K(I-\mathbf{u}^{(m)}\mathbf{u}^{(m)T}). \tag{3.2}$$

Assume that a Krylov iterative method (e.g., CG) is used to solve eq. (2.1) and that no Rayleigh Ritz is performed in the outer JD step, so the method is similar to the unaccelerated Newton method [58]. The latter assumption facilitates an inexpensive way to monitor the eigenvalue convergence inside the linear solver, while being a worst case scenario for the subspace accelerated JD. At the $k-$th inner iteration, the linear system residual is:

$$\mathbf{g}_k = \mathbf{r}^{(m)} + A_{\eta,u^{(m)}}\mathbf{t}_k, \tag{3.3}$$

where $\mathbf{t}_k$ is the current approximate solution to the correction equation. If the linear solver were stopped at the $k-$th step, the corrected eigenvector, its Rayleigh quotient, and its residual would be:

$$\mathbf{u}_k^{(m+1)} = (\mathbf{u}^{(m)} + \mathbf{t}_k)/\|\mathbf{u}^{(m)} + \mathbf{t}_k\| \tag{3.4}$$

$$\theta_k^{(m+1)} = \mathbf{u}_k^{(m+1)T} A \mathbf{u}_k^{(m+1)} \tag{3.5}$$

$$\mathbf{r}_k^{(m+1)} = A\mathbf{u}_k^{(m+1)} - \theta_k^{(m+1)}\mathbf{u}_k^{(m+1)}. \tag{3.6}$$

Note that although $\mathbf{g}_k \to 0$, as $k \to \infty$, the eigenresidual $\mathbf{r}_k^{(m+1)}$ converges not to zero but to the eigenvalue residual of the next INVIT iterate: $\mathbf{r}_\infty^{(m+1)}$. Notay showed that $\|\mathbf{g}_k\|$ and $\|\mathbf{r}_k^{(m+1)}\|$ converge at similar rates up to the point where $\|\mathbf{r}_k^{(m+1)}\|$ approaches $\|\mathbf{r}_\infty^{(m+1)}\|$. Beyond this point, there is no benefit in continuing the inner linear solver and this is the main idea behind JDCG. Notay also developed an inexpensive way to express the norm of the eigenresidual $\|\mathbf{r}_k^{(m+1)}\|$ as a function of the linear system residual $\|\mathbf{g}_k\|$ when CG is the inner solver. When the linear system residual starts to converge at a faster rate than the corresponding eigenvalue residual, the inner iteration is stopped.

Notay showed that for extremal eigenvalues JDCG demonstrates the same convergence as a periodically restarted CG method. Because the number of restarts is small, corresponding to the number of Newton steps rather than to the size of the search space, JDCG achieves near optimality, converging within a small factor of optimal. In section 5.2 we provide intuition on how to quantify this small factor. Notay also showed that if the Ritz value $\theta^{(m)}$ is closer to λ_1 than to λ_2, eq. (2.1) with $\eta = \theta^{(m)}$ is positive definite and CG can be used safely. See [40] for a similar result.

However, positive definiteness of the correction equation can be a limiting factor for a general purpose eigenvalue solver based on JDCG. There are several reasons. First, when good initial guesses are not available, $\theta^{(m)}$ can be well inside the spectrum causing eq. (2.1) to be indefinite. Second, eigensolvers of the GD/JD type can be easily modified to find interior eigenvalues closer to a given shift η. Regardless of the quality of the initial guess, JDCG cannot be used in this case, and we have to rely on methods such as MINRES or SYMMLQ [51]. Third, and perhaps most importantly, the preconditioner K cannot be guaranteed to be positive definite. Practitioners solving linear systems would rarely choose an indefinite preconditioner but for eigenvalue problems good approximations to the shifted matrix $(A - \theta^{(m)}I)$ arise naturally in many applications. Examples include strongly diagonal dominant matrices, for which the Davidson method was originally developed [11], and discretizations of PDEs with spectral or Fourier bases (see experiments in section 5.5). Unfortunately, neither CG nor MINRES can utilize an indefinite preconditioner.

In the following section, we extend the JDCG ideas to the symmetric QMR of Freund and Nachtigal [18, 19] which allows for both indefinite operator and preconditioner. In addition, the monotonic convergence of the linear system quasi-residual allows for more intuitive and efficient stopping criteria than those used in JDCG.

3.2. The JDQMR method. In case of indefinite correction equation or preconditioner, GMRES or BiCGSTAB have been suggested as robust inner solvers [3]. GMRES, however, requires either storage of its whole search space or frequent restarts that degrade its convergence. BiCGSTAB on the other hand does have a short term recurrence, capable of generating a large Krylov space implicitly, but it is oblivious to the symmetry of the matrix and thus twice as expensive per step as CG. Although convergence may be also twice as fast as that of BiCG [72] in the general case, for symmetric matrices this benefit may be smaller as the squared polynomial of BiCGSTAB is clearly suboptimal to the same degree polynomial obtained through an *optimal* method based on the Lanczos recurrence.

3.2.1. Symmetric QMR. The QMR method for non symmetric matrices uses the BiCG [4] short-term recurrence but obtains the approximations by a quasi minimization of the residual over the available terms. Although there is no global optimality property, and the actual residual norm is not minimized, convergence is much smoother than with BiCG. Freund et al. noticed in [18] that when the matrix is symmetric, the simplification of the Lanczos process gives rise to a symmetric version of BiCG that can be used with any symmetric, possibly indefinite preconditioner. Based on this symmetric BiCG, they proposed the symmetric QMR

ALGORITHM 3.1. *Symmetric QMR*

Input: $A_{\eta,u^{(m)}}, K_{u^{(m)}}, -\mathbf{r}^{(m)}, maxiter$

Output: t_k

$\mathbf{t}_0^{(m)} = 0, \delta_0 = 0, \mathbf{r}_0 = -\mathbf{r}^{(m)}, \mathbf{d}_0 = K_{u^{(m)}}^{-1}\mathbf{r}_0$

$\hat{g}_0 = \|\mathbf{r}_0\|, \Theta_0 = 0, \rho_0 = \mathbf{r}_0^T\mathbf{d}_0$

if $(maxiter = 0)$, $\mathbf{t}_0 = \mathbf{d}_0$, **return**

for $k = 1, \ldots, maxiter$

  $\mathbf{w} = A_{\eta,u^{(m)}}\mathbf{d}_{k-1}$

  $\sigma_{k-1} = \mathbf{d}_{k-1}^T\mathbf{w}$, **if** $(\sigma_{k-1} = 0)$, **return**

  $\alpha_{k-1} = \frac{\rho_{k-1}}{\sigma_{k-1}}$

  $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{w}$

  $\Theta_k = \frac{\|\mathbf{r}_k\|}{\hat{g}_{k-1}}, c_k = \frac{1}{\sqrt{1+\Theta_k^2}}, \hat{g}_k = \hat{g}_{k-1}\Theta_k c_k$

  $\delta_k = (c_k^2\Theta_{k-1}^2)\delta_{k-1} + (c_k^2\alpha_{k-1})\mathbf{d}_{k-1}$

  $\mathbf{t}_k = \mathbf{t}_{k-1} + \delta_k$

  if ($\hat{g}_k$ *converged OR* $\rho_{k-1} = 0$), **return**

  $\mathbf{w} = K_{u^{(m)}}^{-1}\mathbf{r}_k, \rho_k = \mathbf{r}_k^T\mathbf{w}, \beta_k = \frac{\rho_k}{\rho_{k-1}}$

  $\mathbf{d}_k = \mathbf{w} + \beta_k\mathbf{d}_{k-1}$

end for

(sQMR) algorithm. sQMR improves the functionality of MINRES, while keeping the same computational requirements. sQMR and MINRES are equivalent without preconditioning.

Assume the correction equation with operator $A_{\eta,u^{(m)}}$, preconditioner $K_{u^{(m)}}$ and the right hand side $-\mathbf{r}^{(m)}$. As it is typical in the JD method, assume that the initial guess is zero. Algorithm 3.1 shows the sQMR algorithm for solving the above correction equation with right preconditioning.

Similarly to CG, the $\mathbf{d}_k$ are the $A_{\eta,u^{(m)}}$ conjugate vectors that BiCG produces, and $\mathbf{r}_k$ are the BiCG residuals of the linear system at the $k-$th step. Hence ([4]):

$$d_i^T A_{\eta,u^{(m)}} d_k = 0, \;\; \forall i = 0, \ldots, k-1 \tag{3.7}$$

$$d_i^T r_k = 0, \;\; \forall i = 0, \ldots, k-1. \tag{3.8}$$

Note that the sQMR residual, $\mathbf{g}_k$, and its norm $g_k = \|\mathbf{g}_k\| = \|\mathbf{r}^{(m)} + A_{\eta,u^{(m)}}\mathbf{t}_k\|$ are *not* computed during the algorithm; only the norm from the quasi minimization $\hat{g}_k$. The actual norm g_k, although typically larger than $\hat{g}_k$, it is bounded by:

$$g_k \leq \sqrt{k+1}\,\hat{g}_k. \tag{3.9}$$

Without preconditioning, $g_k = \hat{g}_k$, and depending on the preconditioner the above bound can be sharp. The actual norm g_k could be computed at the expense of two additional vector updates and one inner product, and by storing an extra vector for $A_{\eta,u^{(m)}}\delta_k$. However, as we show in Section 3.4 this expense is not justified. In the following section, we consider only the actual residual norm g_k.

Finally, note that we have used right preconditioning so that the norm g_k corresponds to the original correction equation, not the preconditioned one (see [18]). This is desirable as we want to relate the eigenvalue residual to the monotonically decreasing linear system residual.

3.2.2. Monitoring the eigenresidual in sQMR. Replacing CG with sQMR results in a method which we call JDQMR. As with JDCG, we must provide expressions for $\theta_k^{(m+1)}$

and $\|\mathbf{r}_k^{(m+1)}\|$ of eqs. (3.4–3.6) that can be calculated through short recurrences in sQMR and without additional matrix-vector, preconditioning, or vector operations. The difference between the expressions for JDCG and JDQMR is strictly due to the projection process. Similarly to [46], we consider a simplified JD process where the Ritz vector is updated through eq. (3.4) and not through a Rayleigh Ritz procedure in the whole search space. A subspace accelerated JDQMR would improve the Ritz value, $\theta_k^{(m+1)}$, but not necessarily the residual norm, $\|\mathbf{r}_k^{(m+1)}\|$. This simplification, therefore, constitutes a worst case scenario for the complete JDQMR, and it also keeps the expressions inexpensive and tractable.

THEOREM 3.1. *Consider Algorithm 3.1 applied to the correction equation (2.1), and the definitions in (3.1–3.6). Subscripts denote the QMR iteration number and superscripts the iteration number of the outer JD process. Based on the notation of the algorithm, define the following scalars:*

$$\mathrm{B}_k = \mathbf{t}_k^T(A-\eta I)\mathbf{u}^{(m)}, \tag{3.10}$$

$$\Gamma_k = \mathbf{t}_k^T(A-\eta I)\mathbf{t}_k = \mathbf{t}_k^T A_{\eta,u^{(m)}}\mathbf{t}_k, \tag{3.11}$$

$$\Delta_k = \delta_k^T\mathbf{r}^{(m)} = -\delta_k^T\mathbf{r}_0, \tag{3.12}$$

$$\Phi_k = \delta_k^T(A-\eta I)\delta_k = \delta_k^T A_{\eta,u^{(m)}}\delta_k, \tag{3.13}$$

$$\Psi_k = \mathbf{t}_{k-1}^T(A-\eta I)\delta_k = \mathbf{t}_{k-1}^T A_{\eta,u^{(m)}}\delta_k, \tag{3.14}$$

$$\gamma_k = c_k^2\Theta_{k-1}^2,\ \xi_k = c_k^2\alpha_{k-1}. \tag{3.15}$$

Then, these expressions hold:

$$\theta_k^{(m+1)} = \eta + \frac{\left(\theta^{(m)}-\eta+2\mathrm{B}_k+\Gamma_k\right)}{1+\|\mathbf{t}_k\|^2} \tag{3.16}$$

$$\|\mathbf{r}_k^{(m+1)}\|^2 = \frac{\|\mathbf{g}_k\|^2}{1+\|\mathbf{t}_k\|^2} + \frac{(\theta^{(m)}-\eta+\mathrm{B}_k)^2}{1+\|\mathbf{t}_k\|^2} - (\theta_k^{(m+1)}-\eta)^2, \tag{3.17}$$

where B_k *and* Γ_k *satisfy the following recurrences:*

$$\Delta_k = \gamma_k\Delta_{k-1} - \xi_k\rho_{k-1} \tag{3.18}$$

$$\mathrm{B}_k = \mathrm{B}_{k-1} + \Delta_k \tag{3.19}$$

$$\Gamma_k = \Gamma_{k-1} + 2\Psi_k + \Phi_k \tag{3.20}$$

$$\Phi_k = \gamma_k^2\Phi_{k-1} + \xi_k^2\sigma_{k-1} \tag{3.21}$$

$$\Psi_k = \gamma_k\Psi_{k-1} + \gamma_k\Phi_{k-1}, \tag{3.22}$$

with $\mathrm{B}_0 = \Delta_0 = \Gamma_0 = \Phi_0 = \Psi_0 = 0$.

Proof. From eqs. (3.4–3.5), $\theta^{(m)} = \mathbf{u}^{(m)T}A\mathbf{u}^{(m)}$, $\|\mathbf{u}^{(m)}\| = 1$, the symmetry of A, and the fact that $\mathbf{t}_k^T\mathbf{u}^{(m)} = 0$, we obtain $\|\mathbf{u}^{(m)}+\mathbf{t}_k\|^2 = 1+\|\mathbf{t}_k\|^2$ and eq.(3.16):

$$\begin{aligned}\theta_k^{(m+1)} &= (\mathbf{u}^{(m)}+\mathbf{t}_k)^T A(\mathbf{u}^{(m)}+\mathbf{t}_k)/(1+\|\mathbf{t}_k\|^2)\\ &= \eta + (\theta^{(m)}-\eta+2\mathbf{t}_k^T(A-\eta I)\mathbf{u}^{(m)}+\mathbf{t}_k^T A_{\eta,u^{(m)}}\mathbf{t}_k)/(1+\|\mathbf{t}_k\|^2).\end{aligned}$$

Because $\mathbf{r}_k^{(m+1)} \perp \mathbf{u}_k^{(m+1)}$ we derive eq.(3.17) as in eqs. (23–25) from [46]:

$$\begin{aligned}\|\mathbf{r}_k^{(m+1)}\|^2 &= \|(A-\eta I)(\mathbf{u}^{(m)}+\mathbf{t}_k)\|^2/(1+\|\mathbf{t}_k\|^2) - (\theta_k^{(m+1)}-\eta)^2\\ &= \Big(\|(I-\mathbf{u}^{(m)}\mathbf{u}^{(m)T})(A-\eta I)(\mathbf{u}^{(m)}+\mathbf{t}_k)\|^2 +\end{aligned}$$

$$\|\mathbf{u}^{(m)}\mathbf{u}^{(m)T}(A-\eta I)(\mathbf{u}^{(m)}+\mathbf{t}_k)\|^2\Big)/(1+\|\mathbf{t}_k\|^2)-(\theta_k^{(m+1)}-\eta)^2$$
$$=(\|\mathbf{g}_k\|^2+(\theta^{(m)}-\eta+\mathrm{B}_k)^2)/(1+\|\mathbf{t}_k\|^2)-(\theta_k^{(m+1)}-\eta)^2.$$

The recurrences are obtained through the vector update formulas in sQMR.

$$\begin{aligned}\mathrm{B}_k &= \mathbf{t}_k^T(A-\eta I)\mathbf{u}^{(m)} = \mathbf{t}_k^T(A-\theta^{(m)}I)\mathbf{u}^{(m)} = \mathbf{t}_k^T\mathbf{r}^{(m)}\\ &= (\mathbf{t}_{k-1}+\delta_k)^T\mathbf{r}^{(m)} = \mathrm{B}_{k-1}+\Delta_k,\\ \Delta_k &= \delta_k^T\mathbf{r}^{(m)} = (\gamma_k\delta_{k-1}+\xi_k\mathbf{d}_{k-1})^T\mathbf{r}^{(m)} = \gamma_k\Delta_{k-1}+\xi_k(\mathbf{d}_{k-1}^T\mathbf{r}^{(m)}).\end{aligned}$$

Notice that $\mathbf{r}^{(m)} = -\mathbf{r}_0 = -\mathbf{r}_{k-1} - A_{\eta,u^{(m)}}\mathbf{t}_{k-1}^{bcg}$, where $\mathbf{r}_{k-1}, \mathbf{t}_{k-1}^{bcg}$ are the residual and the solution that the BiCG process produces at the $k-1$ inner step. We know that the BiCG solution $\mathbf{t}_{k-1}^{bcg}$ is a linear combination of the first $\mathbf{d}_{i,i=0:k-2}$ BiCG vectors, and from eq. (3.7) we have $\mathbf{d}_{k-1}^T A_{\eta,u^{(m)}}\mathbf{t}_{k-1}^{bcg} = 0$. Thus, using the update formula of $\mathbf{d}_{k-1}$ in sQMR and eq. (3.8), we obtain eq. (3.18):

$$\begin{aligned}\Delta_k &= \gamma_k\Delta_{k-1} - \xi_k\mathbf{d}_{k-1}^T\mathbf{r}_{k-1} = \gamma_k\Delta_{k-1} - \xi_k(\mathbf{r}_{k-1}^T K_{u^{(m)}}^{-1}\mathbf{r}_{k-1} + \beta_{k-1}\mathbf{d}_{k-2}^T\mathbf{r}_{k-1})\\ &= \gamma_k\Delta_{k-1} - \xi_k\mathbf{r}_{k-1}^T K_{u^{(m)}}^{-1}\mathbf{r}_{k-1} = \gamma_k\Delta_{k-1} - \xi_k\rho_{k-1}\end{aligned}$$

The recurrence for Γ_k is decomposed as follows:

$$\Gamma_k = (\mathbf{t}_{k-1}+\delta_k)^T A_{\eta,u^{(m)}}(\mathbf{t}_{k-1}+\delta_k) = \Gamma_{k-1}+2\Psi_k+\Phi_k$$

We first consider Φ_k. Notice that from the sQMR algorithm that δ_k is a linear combination of the $\mathbf{d}_{i,i=0:k-1}$. Therefore δ_i are also conjugate to $\mathbf{d}_i$, i.e.,

$$\delta_{k-1}^T A_{\eta,u^{(m)}}\mathbf{d}_{k-1} = 0. \tag{3.23}$$

Then, we can derive eq. (3.21) as follows:

$$\begin{aligned}\Phi_k &= (\gamma_k\delta_{k-1}+\xi_k\mathbf{d}_{k-1})^T A_{\eta,u^{(m)}}(\gamma_k\delta_{k-1}+\xi_k\mathbf{d}_{k-1})\\ &= \gamma_k^2\Phi_{k-1} + \xi_k^2\mathbf{d}_{k-1}^T A_{\eta,u^{(m)}}\mathbf{d}_{k-1} = \gamma_k^2\Phi_{k-1}+\xi_k^2\sigma_{k-1}.\end{aligned}$$

Now, we consider Ψ_k. We observe that the sQMR solution vector $\mathbf{t}_k$ is a linear combination of the $\delta_{i,i=0:k}$ vectors. Using the conjugacy of δ_i with $\mathbf{d}_i$ in eq. (3.23), we have $\mathbf{t}_k^T A_{\eta,u^{(m)}}\mathbf{d}_k = 0$. Then eq. (3.22) is obtained as:

$$\begin{aligned}\Psi_k &= \mathbf{t}_{k-1}^T A_{\eta,u^{(m)}}(\gamma_k\delta_{k-1}+\xi_k\mathbf{d}_{k-1}) = \gamma_k\mathbf{t}_{k-1}^T A_{\eta,u^{(m)}}\delta_{k-1}\\ &= \gamma_k\mathbf{t}_{k-2}^T A_{\eta,u^{(m)}}\delta_{k-1} + \gamma_k\delta_{k-1}^T A_{\eta,u^{(m)}}\delta_{k-1} = \gamma_k(\Psi_{k-1}+\Phi_{k-1}).\end{aligned}$$

Finally, it can be simply verified that for $k=0$ the above scalar quantities have the value of zero. □

3.3. Stopping criteria. The above theorem provides the means to dynamically and inexpensively monitor the eigenvector convergence inside the linear solver. It also characterizes the convergence behavior of both the linear system and the eigenvalue residual. We first consider the unpreconditioned case, where the sQMR is equivalent to MINRES, and $g_k = \hat{g}_k$. To simplify the discussion we introduce the following notation where non-boldface letters are used to denote the norms of the vectors with corresponding bold-face letters. Specifically,

$$r_k^{(m+1)} = \|\mathbf{r}_k^{(m+1)}\| \tag{3.24}$$

$$r_\infty^{(m+1)} = \|\mathbf{r}_\infty\| = \|\lim_{k\to\infty} \mathbf{r}_k^{(m+1)}\| \tag{3.25}$$

$$g_k = \|\mathbf{g}_k\| \tag{3.26}$$

$$t_k = \|\mathbf{t}_k\| \tag{3.27}$$

$$p_k = \frac{(\theta^{(m)} - \eta + \mathrm{B}_k)^2}{1 + \|\mathbf{t}_k\|^2} - (\theta_k^{(m+1)} - \eta)^2. \tag{3.28}$$

Equation (3.17) suggests that as the linear system is solved, $g_k \to 0$, $r_k^{(m+1)} \to r_\infty^{(m+1)}$, and $p_k \to (r_\infty^{(m+1)})^2$. Note that p_k could be negative initially, but converges to a positive limit with rate similar to that of g_k. In addition, Notay showed for a similar equation for the CG that p_k is much smaller than $(r_k^{(m+1)})^2$. Therefore, $r_k^{(m+1)}$ stays close to g_k until the proximity of the limit. Notay proposed to stop the inner solver when convergence rates start to deviate: $g_k/g_{k-1} < (r_k^{(m+1)}/r_{k-1}^{(m+1)})^{\alpha'}$, for some user defined α', say 0.9. The lack of monotonic convergence of g_k in CG complicates the above stopping criterion, and the practical JDCG algorithm introduces several different tests and user defined parameters. The smooth convergence of the JDQMR residuals allows for a much more accurate use of convergence rates. Moreover, we show how to provide a set of almost parameter-free, adaptive stopping criteria.

First rewrite eq. (3.17) as

$$r_k^{(m+1)2} = \frac{g_k^2}{1 + t_k^2} + p_k. \tag{3.29}$$

We want to quantify how accurate an approximation of the INVIT iterate we should obtain, or equivalently to identify an α, such that we stop when

$$r_k^{(m+1)} \leq \alpha\, r_\infty^{(m+1)}, \text{ with } \alpha > 1. \tag{3.30}$$

Based on our discussion above, the results in [46], and the fact that in the domain of Newton convergence (for the outer iteration), $r_\infty^{(m+1)}$ is much smaller than $r_0^{(m+1)}$, we can assume that $p_k \approx r_\infty^{(m+1)2}$ even for relatively small k, and in particular for k close to the required stopping point. Thus, we can turn condition (3.30) into a testable condition involving only g_k and $r_k^{(m+1)}$:

$$\begin{gathered} r_k^{(m+1)2} \leq \alpha^2\, p_k = \alpha^2 r_k^{(m+1)2} - \alpha^2 g_k^2/(1+t_k^2) \Leftrightarrow \\ g_k \leq r_k^{(m+1)} \sqrt{\frac{\alpha^2 - 1}{\alpha^2}} \sqrt{1 + t_k^2}. \end{gathered} \tag{3.31}$$

Because the norm of the correction is typically $t_k \ll 1$, especially close to convergence, the above condition is determined by the factor based on α. A first approach to obtaining this factor is to observe that when the two terms $g_k^2/(1+t_k^2)$ and p_k in eq. (3.29) become equal, $r_k^{(m+1)}$ is approximately within a factor of $\sqrt{2}$ from its limit. Thus, we could stop if the linear system residual is smaller than $0.707\sqrt{1+t_k^2}$ of the eigenresidual. Our experiments, however, have shown this not to be an effective factor. Indeed, a factor close to $0.99\sqrt{1+t_k^2}$ has yielded consistently better results.

To identify an appropriate factor α, we should first understand its effects on the global convergence of the method. On exit from the inner iteration, the eigenvalue errors satisfy the Bauer Fike bounds:

$$|\theta_k^{(m+1)} - \lambda_1| \le r_k^{(m+1)} \tag{3.32}$$

$$|\theta_\infty^{(m+1)} - \lambda_1| \le r_\infty^{(m+1)}. \tag{3.33}$$

We avoid using the tighter bounds of [52] because λ_1 may not be sufficiently separated from the spectrum relative to $r_k^{(m+1)}$. After completing the following outer step, exact INVIT will reduce the eigenvalue error by a factor f, i.e, $fr_\infty^{(m+1)}$. Therefore, a small enough value of α — obviously smaller than $1/f$ — is needed for the inexact INVIT to guarantee convergence. An analysis of the bounds that such a value has to satisfy and some heuristics are presented in [62] (see also [6]). Similarly, exact RQI will yield an eigenvalue error $f|\theta_\infty^{(m+1)} - \lambda_1|^2$, for some factor f. As Smit et al. derive in [62], quadratic convergence in inexact RQI is maintained even for relatively large values of α, especially during the later stages of the iteration, as long as sufficiently small ones are used in the first few iterations.

In the first few outer iterations, the JDQMR may not have a good estimate of the eigenvalue, suggesting the contrary, i.e., that the correction equation should not be solved accurately at all. This presents no problem, because JDQMR is more flexible than INVIT/RQI, in that even if no accuracy is required in the solution of the correction equation (zero inner iterations) the subspace accelerated algorithm still converges as fast as Arnoldi or GD (with preconditioning). The outer step, however, can be rather expensive compared with the recurrence based inner solver. For this reason, it is beneficial to postpone exiting from the QMR as much as possible to achieve an outer convergence similar to that of the exact INVIT/RQI. This implies that $r_k^{(m+1)}$ and $r_\infty^{(m+1)}$ should be of the same order of magnitude, e.g., for INVIT $\alpha < \min(1/f, 10)$. We have chosen $\alpha = 7$, which yields the parameter 0.99 of the previous paragraph. The choice of α that gives an optimal total number of inner iterations, i.e., number of matrix-vector operations, is beyond the scope of this paper and the focus of current research. We note, however, that preliminary results with an RQI model based on equations (3.29) and (3.31) have confirmed that when a constant α is to be used its value should be 8, which is close to our experimental choice.

The criterion (3.31) can be complemented for the case where the convergence of the linear system solver is very slow or experiences plateaus. In that case, we want $r_k^{(m+1)}$ to be even closer to g_k relative to the convergence rate of g_k and not only by a constant factor. We demand, therefore, that the average convergence rate of the eigenresidual $(r_k^{(m+1)}/r_0)^{1/k} = (r_k^{(m+1)}/g_0)^{1/k}$ is no more (i.e., not slower) than the geometric average of the last two average rates of the linear system, i.e., stop when,

$$\begin{aligned} \sqrt{\frac{g_k}{g_0}\frac{g_{k-1}}{g_0}} &< \frac{r_k^{(m+1)}}{g_0} \Leftrightarrow \\ g_k &< r_k^{(m+1)}\sqrt{g_k/g_{k-1}}. \end{aligned} \tag{3.34}$$

Putting together inequalities (3.31) and (3.34) we obtain our first stopping criterion:

$$g_k \le r_k^{(m+1)} \max\left(0.99\sqrt{1+t_k^2}, \sqrt{g_k/g_{k-1}}\right). \tag{3.35}$$

As we discuss in section 5.2, despite the optimality of the above criteria, sQMR may experience convergence plateaus in more than one outer JD iteration. During these plateaus it attempts to resolve certain eigencomponents that prevent convergence for the specific correction equation. Although this plateau would be built even by our QMRopt benchmark, repeating it every outer iteration would be wasteful; wasteful in a different sense than what we have studied till now. The g_k and $r_k^{(m+1)}$ may still be close and the inner iteration may still offer improvements to the correction, but to do so it needs components that it keeps rebuilding for different correction equations at every outer step. Although not severe, this is sometimes observed in problems with slow convergence. In such cases, we have found it beneficial to stop when r_0 is reduced by a certain factor, say

$$r_k^{(m+1)} < 0.1\ r_0. \tag{3.36}$$

Note that this is very different from the traditional $g_k < 0.1\ g_0$, and that it should be used in conjunction with the criteria that guarantee closeness of g_k and $r_k^{(m+1)}$.

For extreme eigenvalues, the variational principle guarantees monotonic convergence of the Ritz value at the outer step, but not of the approximation $\theta_k^{(m+1)}$ during the execution of QMR. We have found it beneficial, however, to stop QMR when

$$\theta_k^{(m+1)} > \theta_{k-1}^{(m+1)}. \tag{3.37}$$

If $\theta_k^{(m+1)}$ increases, an eigenvalue was missed and the solver is trying to retarget convergence to the missed one. It is thus beneficial to exit the inner iteration and let the outer step improve on the new Ritz value through the Rayleigh Ritz procedure, providing a better correction equation during the next inner solution.

Finally, we should stop the inner iteration when the required tolerance has been achieved (or the maximum number of iterations reached). Although $g_k < r_k^{(m+1)}$, except possibly for a few initial steps, it is safer to check both g_k and $r_k^{(m+1)} < \varepsilon_{inn}$. The threshold for the inner iterations, ε_{inn}, could be the same as the user defined threshold, ε, for the eigenresiduals. The $r_k^{(m+1)}$, however, is the residual norm of the corrected Ritz vector, and not from the outer Rayleigh Ritz method on the basis V that includes the correction. Thus the residual norm computed by the outer method could be slightly larger than ε, causing the inner method to be called again to provide a very small correction. This restarting could force QMR to build again certain vector components, wasting some iterations. Therefore, we suggest $\varepsilon_{inn} = \varepsilon/2$, while making sure that this is achievable within the machine's precision, $\varepsilon_{\text{machine}}$. Estimating the norm of A by the largest Ritz value seen thus far, we stop QMR if:

$$g_k < \varepsilon_{inn} \text{ OR } r_k^{(m+1)} < \varepsilon_{inn}, \tag{3.38}$$

$$\text{with } \varepsilon_{inn} \ = \ \max(\varepsilon/2, \varepsilon_{\text{machine}}\|A\|). \tag{3.39}$$

3.4. The effect of quasi-minimization. With a preconditioner K, the BiCG residuals $\mathbf{r}_k$ in the Algorithm 3.1 are K^{-1}-orthogonal, but unlike MINRES which implicitly considers the orthogonal basis $K^{-1/2}\mathbf{r}_k$, sQMR uses the $\mathbf{r}_k$ and thus the actual norm g_k of the sQMR residual is bounded as shown in inequality (3.9). A natural question is whether the above stopping criteria would be adversely affected by using the readily computed $\hat{g}_k$ instead of g_k.

The only quantity in Theorem 3.1 that is affected by replacing g_k with $\hat{g}_k$ is the eigenresidual norm $r_k^{(m+1)}$. Neither the eigenvalue estimate $\theta_k^{(m+1)}$ nor the p_k term of eq. (3.29) depends on the choice of g_k. Let $g_k = c_k\hat{g}_k$, with $1 \le c_k \le \sqrt{k+1}$. We do not consider the possibility $0 \le c_k < 1$, because in the rare case that this occurs, c_k would still be very close to 1. Denote by $\hat{r}_k^{(m+1)}$ the eigenresidual norm approximation computed through eq. (3.29) using $\hat{g}_k$. We can express $r_k^{(m+1)}$ as a function of the approximate $\hat{r}_k^{(m+1)2}$ as follows:

$$r_k^{(m+1)2} = \frac{g_k^2}{1+t_k^2} + p_k = c_k^2 \frac{\hat{g}_k^2}{1+t_k^2} + p_k = c_k^2(\hat{r}_k^{(m+1)2} - p_k) + p_k$$
$$= c_k^2 \hat{r}_k^{(m+1)2} - (c_k^2 - 1)p_k. \tag{3.40}$$

Ideally, criterion (3.35) should check the ratio of the exact residuals $g_k/r_k^{(m+1)}$. Denoting $R = \max(0.99\sqrt{1+t_k^2}, \sqrt{g_k/g_{k-1}})$ and using (3.40) we should check:

$$\frac{g_k^2}{r_k^{(m+1)2}} = \frac{c_k^2 \hat{g}_k^2}{c_k^2 \hat{r}_k^{(m+1)2} - (c_k^2-1)p_k} = \frac{\hat{g}_k^2}{\hat{r}_k^{(m+1)2}} \frac{1}{1 - \frac{(c_k^2-1)}{c_k^2} \frac{p_k}{\hat{r}_k^{(m+1)2}}} < R^2.$$

Equivalently, using our computed residual approximations we should check:

$$\hat{g}_k < \hat{r}_k^{(m+1)} \; R \sqrt{1 - \frac{(c_k^2-1)}{c_k^2} \frac{p_k}{\hat{r}_k^{(m+1)2}}}. \tag{3.41}$$

In practice, because c_k is not known, we can only check:

$$\hat{g}_k < \hat{r}_k^{(m+1)} \; R. \tag{3.42}$$

We have used the same R in both ideal and practical tests, because $\hat{g}_k/\hat{g}_{k-1} = (c_{k-1}/c_k)g_k/g_{k-1} \approx g_k/g_{k-1}$ as c_k can change only slowly between successive steps.

When $\hat{r}_k^{(m+1)}$ is far from convergence, $p_k \ll \hat{r}_k^{(m+1)2}$, and since $(c_k^2-1)/c_k 2 < 1$, the factor multiplying R in (3.41) is almost one, and the approximate criterion (3.42) is practically equivalent to (3.41). Near convergence, $\hat{r}_k^{(m+1)2} \to r_\infty^{(m+1)2} \approx p_k$, so a large c_k could yield a substantially small factor, thus invalidating the approximate criterion. Two issues prevent this.

First, criterion (3.35) was derived from the condition that $r_k^{(m+1)2}$ should not come closer than α^2 to p_k. This gave rise to eqs. (3.30–3.31). Working backwards from R we have $\alpha^2 = 1/(1-R^2)$, so at any time before criterion (3.35) exits, it holds:

$$\frac{p_k}{r_k^{(m+1)}} < 1 - R^2. \tag{3.43}$$

Substituting $r_k^{(m+1)2}$ for $\hat{r}_k^{(m+1)2}$ from eq. (3.40), and using (3.43) we can bound $\frac{p_k}{\hat{r}_k^{(m+1)}}$:

$$\frac{p_k}{\hat{r}_k^{(m+1)}} = \frac{p_k c_k^2}{r_k^{(m+1)} + p_k(c_k^2-1)} < \frac{c_k^2(1-R^2)}{1+(1-R^2)(c_k^2-1)}. \tag{3.44}$$

Then the factor multiplying R in the ideal bound (3.41) is bounded from below:

$$\sqrt{1 - \frac{(c_k^2-1)}{c_k^2} \frac{p_k}{\hat{r}_k^{(m+1)2}}} > \sqrt{\frac{1}{1+(1-R^2)(c_k^2-1)}}. \tag{3.45}$$

The above factor is always very close to 1, so checking (3.42) instead of (3.41) is a good approximation. For example, given that $R \geq 0.99$, it requires an excess of 1000 sQMR iterations, and a c_{1000} close to the bound (3.9) to require a stopping criterion of 0.95 instead of the used 0.99.

The second reason preventing a small factor (3.45) is that the term of $1-R^2 = 1 - g_k/g_{k-1}$ becomes increasingly small with slow linear system convergence. Thus, it compensates for a possible increase in the term $c_k^2 - 1$ that could result from a large number of sQMR iterations, keeping the overall term $1 + (1-R^2)(c_k^2 - 1) \approx 1$.

Finally, even if the factor (3.45) does become slightly less than one, the inner sQMR is stopped a few iterations earlier than if the correct residual norms were used. This, in fact, may be beneficial because no time is wasted overconverging the inner iteration when more outer iterations are needed. On the contrary, using the upper bound $\sqrt{k+1}\,\hat{g}_k$ as the norm estimate requires more iterations to satisfy the stopping criterion, and has consistently underperformed the $\hat{g}_k$ choice in our experiments.

The use of the approximate $\hat{g}_k$ and $\hat{r}_k^{(m+1)}$ may also affect criterion (3.39). Convergence may be signaled a few iterations earlier than it is actually achieved, which could cause an additional correction equation to be solved through a few additional sQMR iterations. However, this does not affect the overall convergence behavior of the algorithm. Moreover, as without preconditioning, the inner tolerance ε_{inn} can be chosen heuristically to reduce the number of unnecessary iterations. For example, the norm ratio $\hat{c}_k = \hat{r}_k^{(m)}/r^{(m)}$ from the previous outer iteration can be used to estimate c_k and converge to $\varepsilon_{inn} = \hat{c}_k\varepsilon$. In our experiments the simple criterion from eq. (3.39) was used and performed consistently well.

3.5. The JDQMR algorithm. Our stopping criteria and the recurrences of Theorem 3.1 can be implemented trivially into the sQMR Algorithm 3.1, which in turn is called at step (19) of Algorithm 2.1. Algorithm 3.2 provides the additional lines required in Algorithm 3.1. Except for the initializations, only line (12) is replaced.

The GD outer algorithm can be implemented with storage for $2m_{max}$ long vectors of size N. If JDQMR is used, there is an additional requirement of 5 long vectors.

Excluding matvec and preconditioning operations, the expense of the outer GD Algorithm 2.1 can be calculated as a function of flops. Following classic literature, each of the steps takes: re-orthogonalization $O(8Nm+2N)$ (step 5), updating of H $O(2mN)$ (step 7), the small eigenproblem $O(4/3m^3)$ (step 8), Ritz vector computation $O(2Nm)$ (step 9), residual computation $O(2Nm+4N)$ (steps 9–10), norm computation $O(2N)$ (step 11), and the restarting cost is $O(4Nm_{max}m_{min})$ (steps 12–19). Averaging over $m = m_{min} \ldots m_{max}$ and setting $\mu = m_{min}/m_{max}$, we have the average cost per step:

$$\text{GDcost} = O\left(\frac{7+4\mu-7\mu^2}{1-\mu} N m_{max} + \frac{13+\mu}{1-\mu} N + 1/3 m_{max}^3\right).$$

Typically, $m_{min} = m_{max}/3$, in which case: $\text{GDouter} = 11.3Nm_{max} + 20N + m_{max}^3/3$. These are mostly BLAS level 2 and level 1 operations. Similarly, we can obtain the cost for each QMR step, including the JD projectors: $\text{QMRcost} = 24N$. However, these are strictly level 1 BLAS operations. The QMR part of JDQMR is about 15% more expensive than the CG part of JDCG, but this is justified by the increased robustness and efficiency.

4. The Conjugate Gradients view. An alternative to Newton is the use of the nonlinear Conjugate Gradient (NLCG) method to minimize the quadratic form of the Rayleigh quotient on the unit ball. As with the Newton approach, the NLCG must be applied on the Grassmann (or Stiefel) manifold for this constrained minimization problem. The appropriate forms of the constrained version for the NLCG and for the case of many required eigenvalues are given in [15]. Researchers have been using similar type of recurrences quite successfully for many decades; see Bradbury and Fletcher's seminal work [8], a long list of references in [15], as well as work in [20, 5].

ALGORITHM 3.2. *JDQMR additions to sQMR algorithm*

(2.1) $\mathrm{B}_0 = \Delta_0 = \Gamma_0 = \Phi_0 = \Psi_0 = 0$

...

(12.0) $\gamma_k = c_k^2 \Theta_{k-1}^2,\ \xi_k = c_k^2 \alpha_{k-1}, f = 1 + \|\mathbf{t}_k\|^2$

(12.1) $\Psi_k = \gamma_k(\Psi_{k-1} + \Phi_{k-1})$

(12.2) $\Phi_k = \gamma_k^2 \Phi_{k-1} + \xi_k^2 \sigma_{k-1}$

(12.3) $\Gamma_k = \Gamma_{k-1} + 2\Psi_k + \Phi_k$

(12.4) $\Delta_k = \gamma_k \Delta_{k-1} - \xi_k \rho_{k-1}$

(12.5) $\mathrm{B}_k = \mathrm{B}_{k-1} + \Delta_k$

(12.6) $p = \left(\theta^{(m)} - \eta + 2\mathrm{B}_k + \Gamma_k\right)/f$

(12.7) $\theta_k^{(m+1)} = \eta + p$

(12.8) $p_k = (\theta^{(m)} - \eta + \mathrm{B}_k)^2/f - p^2$

(12.9) $r_k^{(m+1)} = \sqrt{g_k^2/f + p_k}$

(12.10) **if** *(ρ_{k-1} = 0),* **return**

(12.11) **if** *($r_k^{(m+1)}$ not real),* $r_k^{(m+1)} = \sqrt{g_k^2/f}$

(12.12) **if** *(* $g_k \le r_k^{(m+1)} \max\left(0.99\sqrt{f}, \sqrt{g_k/g_{k-1}}\right)$

OR ($\theta_k^{(m+1)} > \theta_{k-1}^{(m+1)}$ *) OR if desired (*$r_k^{(m+1)} < 0.1 r_0$*)*

OR ($g_k < \varepsilon_{inn}$ *) OR (* $r_k^{(m+1)} < \varepsilon_{inn}$ *))*

then return *the correction* $\mathbf{t}_k$.

NLCG methods build iterates within a Krylov space as they only utilize gradient information. Therefore they cannot converge faster than unrestarted Lanczos or the optimal QMR benchmark. NLCG, however, uses a three term recurrence to compute both eigenvalue and eigenvector without storing all the intermediate iterates. Therefore, NLCG can often find one eigenpair in less time than the Lanczos method despite a larger number of iterations.

Although the relations between NLCG, Newton and quasi Newton methods are well documented [45], the question whether truncated Newton methods are more efficient than NLCG (or quasi Newton) is problem dependent. In the extreme case of a linear unconstrained problem the NLCG is equivalent to CG, and since only one Newton step is required in that case, NLCG and truncated Newton are equivalent. For eigenvalue problems, the nonlinearity is not severe and the above NLCG variants perform quite well. Given that JDQMR is a truncated Newton method with a number of matrix-vector products within a small factor of optimal, we would like to explore its relation to NLCG variants.

4.1. The locally optimal recurrence. Most NLCG approaches consider minimization of the functional along a search direction which is usually conjugate to the previous search direction with respect to some variation of the Hessian. Although not usually expressed like this, the approximation $\mathbf{u}^{(m+1)}$ at the $(m+1)$-th step belongs in the space $L = \left\{\mathbf{u}^{(m-1)}, \mathbf{u}^{(m)}, \mathbf{r}^{(m)}\right\}$, where $\mathbf{r}^{(m)} = A\mathbf{u}^{(m)} - \theta^{(m)}\mathbf{u}^{(m)}$ is the residual of $\mathbf{u}^{(m)}$. It is natural to consider a method that minimizes the Rayleigh quotient on the whole space L instead of only along one search direction. The method:

$$\mathbf{u}^{(m+1)} = \mathrm{RayleighRitz}\left(\{\mathbf{u}^{(m-1)}, \mathbf{u}^{(m)}, \mathbf{r}^{(m)}\}\right),\ m > 1, \tag{4.1}$$

was proposed by D'yakonov in 1983 [14], and was studied further under the name locally optimal Conjugate Gradient (LOCG) in [30]. It is captivating in its simplicity, especially because it avoids the question of how to pick the search directions. Most importantly, the method seems to consistently outperform other NLCG type methods.

Yet, for many years since its inception, LOCG was plagued by numerical instability problems. Obviously, a converging vector $\mathbf{u}^{(m)}$ implies an increasingly linearly dependent basis for the Rayleigh Ritz process. Orthogonalization of the basis was not considered, mainly to keep recurrence costs low, but also because orthogonalizing two almost identical vectors would result primarily in a noise vector. In that case, the effectiveness of the method can be no worse than steepest descent. However, these problems do not arise until a good level of convergence has already been achieved. An extension to this method for many eigenvalues, and with a more stable recurrence (using $\mathbf{u}^{(m)} - \tau^{(m)}\mathbf{u}^{(m-1)}$, for some weight $\tau^{(m)}$, instead of $\mathbf{u}^{(m-1)}$) is the more recently proposed locally optimal block preconditioned CG (LOBPCG) method [32].

4.2. The locally optimal restarting. Restarting Krylov and Krylov-like methods every k iterations can have detrimental effects on their convergence, as the minimization of the Rayleigh quotient occurs only over the last k basis vectors. In the extreme case of $k = 1$ the method is simply the steepest descent. In linear systems of equations, CG has the remarkable property of implicitly remembering all the visited directions. Although the Lanczos method has the same memory, the Lanczos vectors must be revisited to compute the eigenvector. For symmetric problems, a larger subspace acceleration of steepest descent (larger k) yields better convergence, although beyond a certain basis size orthogonalization and restarting costs start to dominate.

Thick restarting [70] is a technique that improves convergence of restarted iterative methods by keeping more than the required Ritz vectors at restart ($m_{min} > nev$). The technique harnesses the superlinear convergence of Lanczos as nearby Ritz vectors are improved, and therefore are gradually deflated. It is theoretically equivalent to Implicit Restarting [63], and it has proved particularly effective when more than one eigenvalue is required. Still, by itself, thick restarting cannot capture the memory of directions that CG so well captures.

In [44], Murray, Davidson and Racine proposed to restart the Davidson method with not only the required Ritz vector at the current step ($\mathbf{u}^{(m_{max})}$), but also with the Ritz vector at the previous step ($\mathbf{u}^{(m_{max}-1)}$). The motivation was to maintain the same three term space L that CG uses to get its optimal approximation when solving a linear system of equations. However, it was unclear what this linear system is, and without the obvious connection to NLCG the method went relatively unnoticed for some years. Also, the suggested implementation was orthogonalization-heavy requiring even extra matrix-vector products. On the other hand, the convergence improvements for Davidson, with or without preconditioning, were impressive.

4.3. The GD+k method. In [68], we showed a theoretical justification of the connection of the above restarting scheme, which we called GD+1, to CG. For completeness we mention the following two pertinent results. Proofs and additional discussion are given in [68, 64].

THEOREM 4.1. *Let vector $\mathbf{u}^{(0)}$, with $\|\mathbf{u}^{(0)}\| = 1$, $\theta^{(0)} = \mathbf{u}^{(0)T}A\mathbf{u}^{(0)}$, and $\eta \in \Re$. Let $(\theta^{(j)}, \mathbf{u}^{(j)})$ be a Ritz pair obtained after j steps of the Lanczos method, with $\mathbf{u}^{(0)}$ as a starting vector. Let $\mathbf{u}_j^{(1)} = \mathbf{u}^{(0)} + \mathbf{t}_j$ be the approximate eigenvector, where $\mathbf{t}_j$ is the correction obtained by applying j steps of CG to the correction equation (2.1) (for $m = 0$). Then:*

$$\mathbf{u}_j^{(1)} = \mathbf{u}^{(j)} \Leftrightarrow \eta = \theta^{(j)}.$$

The theorem says that if we knew the Ritz value $\theta^{(j)}$ at any Lanczos step, we could construct a (CG) three term recurrence to yield the corresponding Ritz vector. Intermediate vector iterates would differ in general. This is similar to the relation between the iterates obtained

ALGORITHM 4.1. *Modifications of Algorithm 2.1 for GD+k restarting*

(7.1)	$s_i^{old} = s_i,\ i = 1,\ldots,m$
(12.1)	*Orthogonalize* $s_i^{old},\ i = 1,\ldots,k$ *among themselves and against* $s_i,\ i = 1,\ldots,m_{min}$
(12.2)	*Compute* $H_{sub} = s^{oldT} H s^{old}$
(12.3)	*Set* $s = [s_1,\ldots,s_{m_{min}}, s_1^{old},\ldots,s_k^{old}]$
(14)	**for** $i = 2,\ldots,m_{min}+k$
(17.1)	$H(m_{min}+1 : m_{min}+k, m_{min}+1 : m_{min}+k) = H_{sub}$
(17.2)	$m = m_{min}+k$

by the optimal QMR process ($\eta = \lambda$) and the Ritz vectors of the unrestarted Lanczos. The effectiveness of the GD+1 restarting technique is attributed to the fact that the Lanczos Ritz vectors approximate the CG iterates as $\theta^{(j)}$ converges. For the smallest eigenvalue λ_1 we have:

LEMMA 4.2. *Let $\gamma = \lambda_2 - \lambda_1$ the gap between the two smallest eigenvalues. If $|\theta^{(0)} - \lambda_1| < \delta$, for $\delta < \gamma/2$, then the distance between the Lanczos Ritz vectors and the approximate eigenvectors produced by CG on eq. (2.1) is bounded by:*

$$\frac{\|\mathbf{u}^{(j)} - \mathbf{u}_j^{(1)}\|}{\|\mathbf{u}_j^{(1)}\|} \leq \frac{|\theta^{(j)} - \eta|}{\gamma - 2\delta}.$$

If we were to restart GD at the m_{max} step, the Ritz value at the $(m_{max}+1)$ step would be minimum over only $\mathbf{u}^{(m_{max})}$ and $\mathbf{r}^{(m_{max})}$. If we also kept $\mathbf{u}^{(m_{max}-1)}$, the three vector subspace would be similar to the three term CG recurrence subspace that yields the exact $\mathbf{u}^{(m_{max}+1)}$. *In fact, the lemma suggests that the distance between the unrestarted and the GD+1 Ritz vectors at the $(m_{max}+1)$ step is bounded by $O(|\theta^{(m_{max}-1)} - \theta^{(m_{max}+1)}|)$.* This justifies a stronger local optimality of the GD+1, not only with respect to the Rayleigh quotient as in LOCG, but also to the Ritz vectors.

In [68], we extended GD+1 to GD+k and combined it with thick restarting. By GD(m_{min}, m_{max})+k we denote the GD method with basis size m_{max}, where at restart we retain m_{min} smallest Ritz vectors from step m_{max} and in addition k smallest Ritz vectors from step $m_{max}-1$. The special case GD(1,3)+1 is mathematically equivalent to LOCG (and in general a block GD($k,3k$)+k with block size k is equivalent to LOBPCG). We also provided an implementation that requires no additional orthogonalization of long vectors and no extra matrix-vector products, by working exclusively with the Rayleigh Ritz coefficients. The modifications required to the GD Algorithm 2.1 are shown in Algorithm 4.1. The expense in terms of flops of steps (12.1–12.3) is $O(m_{max}k^2)$ and thus insignificant for large problems. The only additional expense over GD is that we have to restart with a basis size of $m_{min}+k$ (step 14). Because convergence improves significantly, even with $k = 1$, a much smaller m_{min} can be used than regular GD, and thus GD+1 can be less expensive per step than GD.

We emphasize that the GD+k stability stems from the orthogonality of the basis V. Typically, the small coefficient vectors s_1 and s_1^{old} will be very close. Orthogonalizing s_1^{old} against s_1 will produce a vector that may not lie exactly in the span of s_1^{old}, but it will be orthogonal to s_1. This orthogonality is bequeathed to Vs (as $V^TV = I$), which becomes a stable basis for the LOCG recurrence. In contrast, the LOBPCG method considers the vector $V(s_1 - s_1^{old})$ (with $V^TV \neq I$) which is not exactly orthogonal to Vs_1, and so the Rayleigh Ritz procedure gives rise to a generalized eigenvalue problem, with inferior stability properties. The dangers of a non-orthogonal basis in LOBPCG have been pointed out in [28].

4.4. Global optimality and subspace acceleration. Despite the local optimality of GD+1, from both the Ritz value and Ritz vector viewpoints, there is no theoretical result showing that GD(1,3)+1 (or equivalently LOCG) is within a small factor of optimal QMR. Extensive numerical experiments in [68, 64, 32] and in this paper suggest that this is true for the majority of cases. The above theory can provide some intuition for this. Because the eigenvalue error in symmetric problems converges as the square of the residual norm, the Ritz value quickly approximates the eigenvalue and therefore the LOCG recurrence approximates well the optimal CG method (i.e., solving eq. (2.1) with $\eta = \lambda_1$). Interestingly, even when the Ritz value nearly stagnates over many steps, Lemma 4.2 suggests that the LOCG recurrence can effectively built a space similar to the unrestarted Lanczos. When the algorithm realizes that it was targeting the wrong eigenvalue, the global optimality of the algorithm is lost. This is similar to terminating the inner iteration of the JDQMR to update the shift in the correction equation. Unlike JDQMR, however, each step of GD+1 involves the most current Ritz value, which may explain its faster observed convergence.

Yet, for some spectrum distributions or adversely chosen initial guesses, the Ritz value may vary substantially, often locked temporarily on various interior eigenvalues. The result is several breaks in the global optimality of LOCG, similarly to many outer steps of a JDQMR (or RQI) algorithm. For such cases subspace acceleration and thicker restarting can improve convergence dramatically. The GD(m_{min}, m_{max})+1 method takes advantage of the global optimality when this is present, and when not, it maintains the subspace convergence of the restarted GD/Lanczos for required and nearby eigenvalues. This in turn makes the Ritz value converge faster, thus expediting the return to benefiting from the LOCG recurrence. See also [50] for the effects of subspace acceleration on classic GD. Finally, although GD(m_{min}, m_{max})+1 is a stable implementation, convergence of the coefficient vectors s_1 and s_1^{old} in Algorithm 4.1 yields a basis V which, although orthogonal, may not fully correspond to the ideal three term space of the LOCG recurrence. Without subspace acceleration, GD(1,3)+1 or LOBPCG revert back to steepest descent in these cases.

There is an interesting analog from the nonlinear viewpoint. NLCG is close to optimal when the nonlinearity of the problem is not severe. When it is, e.g., when the minimization function gets trapped in various saddle points (interior eigenvalues), quasi-Newton methods almost always converge faster than NLCG [24]. These methods use the vector iterates to construct incrementally an approximation to the Hessian. A less expensive, but equally effective in terms of convergence, alternative is to use only the last few vector iterates as in the popular L-BFGS method, which is a limited memory version of the Broyden, Fletcher, Goldfarb, and Shanno method [45]. In the context of eigenvalue problems, implementing L-BFGS is similar to the subspace acceleration of the restarted GD+1 method; the only difference being the minimization occurring over lines instead of the whole subspace in GD. At one extreme, L-BFGS with memory 1 (in addition to the current iterate) is equivalent to NLCG and similar to GD(1,3)+1, while at the other extreme unlimited memory BFGS is similar to unrestarted GD (or Arnoldi). In the case of a strictly convex, quadratic objective function the equivalence extends between certain forms of L-BFGS, NLCG and the usual CG [35]. Similar connections between quasi Newton methods and iterative methods for linear systems have been studied. For example, the Broyden method for linear systems gives rise to the EN method which is equivalent to GMRES [16]. In conclusion, we expect the subspace accelerated GD(m_{min}, m_{max})+1 to be more robust and more effective than simple recurrences (such as LOBPCG).

5. Experiments. Our first goal in the experiments is to verify that JDQMR and GD+1 are indeed within a small factor of the optimal method QMRopt. QMRopt refers to the JDQMR method where the exact eigenvalue is given as shift. Thus, one JDQMR outer step, with QMR solving eq. (2.1) resolves the smallest eigenpair. Our second goal is to show the robustness and consistency of the JDQMR and GD+1 algorithms in a variety of situations, including also interior eigenvalue problems. Third, we provide extensive comparisons with JDCG, JDQR, ARPACK, LOBPCG, JDBSYM, and various versions of Shift-Invert Lanczos, with and without preconditioning. Fourth, as JDQMR and GD+1 arise as the methods of choice, we assess their relative merits. This helps a dynamic choice between the two methods, as they both share the same implementation, leading to an overall nearly optimal method.

5.1. Tests and environment. Most of the algorithms used in our experiments are implemented in Matlab. Our JDQMR/GD+1 code implements Algorithms 2.1-3.2 relatively closely, by modifying the JDCG code of [46], which in turn is a modification of the code in [17]. To compare with ARPACK we use the Matlab built-in function `eigs()`, and for LOBPCG we use Revision: 4.0, Beta 4, with the modification that `eig(A,B)` is used instead of `eig(A,B,'chol')`, for numerical stability. We also provide a set of experiments with three packages written in C: our PRIMME multimethod package that implements also GD+1 and JDQMR[1], BLOPEX which is a LOBPCG implementation [29], and JDBSYM, a block Jacobi Davidson implementation [22].

In all experiments we seek only one, smallest algebraic eigenpair. For GD based methods (GD+1, JDCG, JDQMR, JDQR) we use $m_{min} = 6$ and $m_{max} = 18$. With eigs() we select a basis size of 36 so that it requires the same storage as GD+1. LOBPCG requires storage for 6 vectors. The SI-eigs, implements the shift-invert Lanczos by calling QMR to perform the inversions at each step of eigs(). A shift very close to the eigenvalue ensures that only a few outer eigs steps are needed. Because for LOBPCG we have found it far more stable to scale the matrix first, we always use $A = A/\|A\|_F$, where $\|A\|_F$ is the Frobenious norm of the matrix. Then we iterate all methods until the residual norm falls below 10^{-15}. The tolerance for eigs() is also set to 10^{-15}. All methods start with the same random initial guess. When there is a preconditioner, this is always the Matlab function `cholinc(A+sI,1e-3)`, except for NASASRB for which the threshold is `1e-5`. The shift `s` is chosen for each matrix so that the incomplete Choleski factorization can be carried out stably. Experiments are run in Matlab 7 (R14SP3) on an Apple G5 with 1 GB of memory and two 2GHz processors, each with 512 MB L2 cache. For the C experiments we do not scale the matrix, but converge until the residual norm falls below $\|A\|_F 10^{-15}$. Experiments are run on the same Apple G5, using the gcc-4.0.0 compiler with -O flags, but without optimized BLAS/LAPACK libraries.

One of our goals is to provide experiments that can be confirmed independently by other researchers, but are also representative of various classes of problems. Thus, we have selected thirteen different matrix problems available in the following repositories: Matrix Market, University of Florida, and the FEAP collection. Two of these matrices (LUNDA and NASASRB) stem from eigenvalue computations, while the rest have spectra that present various levels of difficulty to iterative methods and the preconditioner. In addition, we have selected three matrices from eigenvalue applications, two from vibrational analysis of molecular structures [75], and the usual five point Laplacian operator on the unit cube. A larger five point Laplacian is used for the C experiments. Our test problems cover the range from easy to difficult, from small to relatively large, and from sparse to relatively dense. Table 5.1 shows the name, sizes, and source for each matrix.

5.2. Nearly optimal convergence. First, we present results from matrices 1138BUS and NASASRB, as they are quite representative and demonstrate both the near optimality and robustness of GD+1 and JDQMR. Figures 5.1 and 5.2 show, for 1138BUS and NASASRB

---
[1] Available at `http//www.cs.wm.edu/~andreas/software`

TABLE 5.1

The matrices used in the experiments. Most matrices come from Matrix Market (MM) [7], the University of Florida matrix repository (UF) maintained by Tim Davis [12], and the FEAP collection by Mark Adams [1]. SPARSKIT (SKIT) [54] was used to generate a 5-point finite difference Laplacian on the unit 3-dimensional cube with Dirichlet conditions. nd3kf and or56f are difficult eigenvalue problems from [75].

No Matrix	N	NNZ	Source	No Matrix	N	NNZ	Source
1. 1138BUS	1138	4054	MM	9. nd3kf	9000	3279690	Yang
2. BCSSTK09	1083	18437	MM	10. or56f	9000	2890150	Yang
3. LUNDA	147	2449	MM	11. Fillet13K_A	13572	632146	FEAP
4. 494BUS	494	1666	MM	12. Cone_A	22032	1433068	FEAP
5. 685BUS	685	3249	MM	13. Plate33K_A0	39366	914116	FEAP
6. BCSSTK16	4884	290378	MM	14. Wing22K_A	22266	923922	FEAP
7. cfd1	70656	1825580	UF	15. Laplacian3d	27000	189000	SKIT
8. finan512	74752	596992	UF	16. NASASRB	54870	2677324	MM

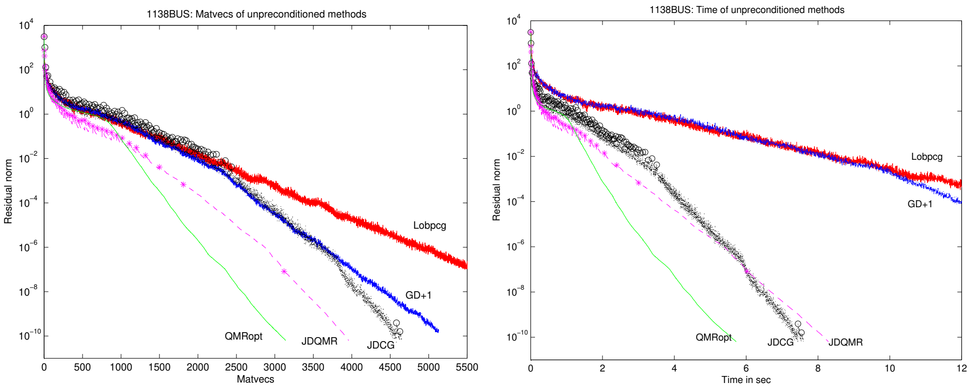

FIG. 5.1. *Residual convergence history for various methods without preconditioning for matrix 1138BUS (see Table 5.1). (∗) and (○) designate the residual norms at outer steps of JDQMR and JDCG respectively. JDQMR improves convergence over JDCG, and GD+1, while being within a factor of 1.26 of the optimal QMR method. Because of the very inexpensive matrix-vector operation for 1138BUS, the convergence advantages of JDQMR are amortized by a slightly more expensive inner iteration over JDCG. This observation is amplified for the GD+1, which improves only on LOBPCG. eigs and SI-eigs require about 40000 matvecs and 18 seconds each.*

respectively, the convergence history of the residual norm for several unpreconditioned methods as a function of matvec operations (left graphs) and as a function of time (right graphs). As expected, the QMRopt is the fastest method both in matvecs and in actual time, since it basically solves one linear system with QMR. Nevertheless, both JDQMR and GD+1 are very close to QMRopt.

For matrix 1138BUS, QMRopt displays superlinear convergence after about 800 matvecs, which is expected despite the highly clustered eigenvalues as the matrix is of small dimension. On the other extreme, LOBPCG achieves a linear convergence similar to the initial phase of QMRopt, but it never switches to superlinear convergence. Despite the limited basis size of 18, GD+1 manages to capture enough spectral characteristics and converges much faster than LOBPCG and similar to JDCG. The importance of the +1 restarting scheme is notable: a thick restarted, larger basis eigs() takes 38376 matvecs (7 times more than GD+1) to converge. Note also that the time for GD+1 is only slightly faster than LOBPCG because the matrix is extremely sparse and minimizing the number of matvecs does not substantially outweigh the more expensive iteration step. Similarly, JDQMR convergence is faster than JDCG, but their times are similar. Both JDQMR and JDCG convergence curves are very close to the QMRopt one.

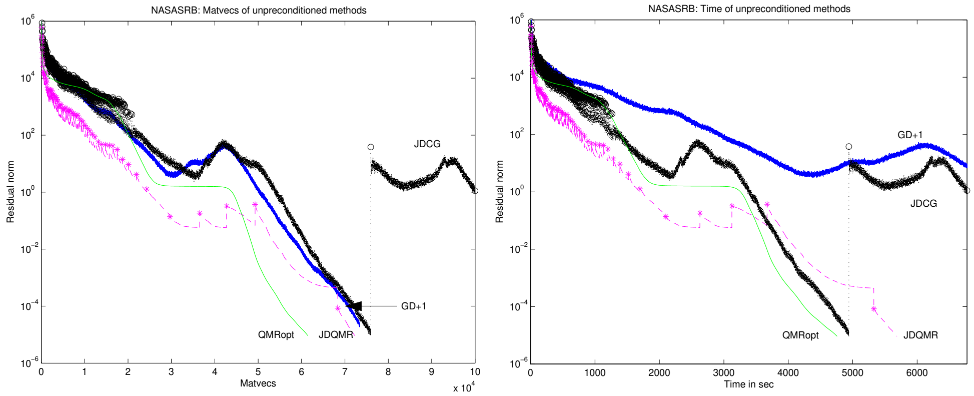

FIG. 5.2. *Residual convergence history for various methods without preconditioning for matrix NASASRB. The smallest eigenvalues are extremely hard to find. eigs seems to stagnate after 300000 matvecs. JDQMR is within 1.17 of the optimal QMR, and so is GD+1. JDCG converges similarly to GD+1 up to 76000 matvecs, at which point the outer step realizes that the correction vector did not correspond to the lowest eigenvalue. We have noticed this behavior also on other matrices. A relatively inexpensive matrix-vector operation plagues the time of GD+1. Convergence for LOBPCG was far slower and is not shown.*

For matrix NASASRB, LOBPCG does not converge fast enough, and eigs() stagnates after a large number of steps. Yet, the convergence of GD+1, JDCG, and JDQMR is almost identical to the one of QMRopt! With one exception: JDCG has been converging to the second lowest eigenvalue, and only realized it near convergence. At that point, JDCG retargets and convergence is delayed accordingly. Besides slower convergence, this behavior may cause misidentification of eigenvalues for larger tolerances. We have noticed this JDCG behavior in a few of the other matrices we have studied. Our stopping criteria are apparently more robust in this direction making JDQMR the fastest method (and closest to QMRopt) for this matrix.

Figure 5.3 shows a summary of our results from applying several methods on all the 16 matrices without preconditioning. For each matrix, the bars show the ratios of matvecs and time (top and lower graph respectively) taken by each method over the QMRopt method. For completeness, we also provide the actual matvecs taken by QMRopt, GD+1, and eigs in Table 5.2. The results are surprisingly consistent, and they reflect also our experience with a larger number of different test problems. The main conclusion is that truncated Newton methods (JDCG/JDQMR) are much more efficient in finding one eigenvalue than eigs() (and much more so than LOBPCG) *even without preconditioning*. This goes against the common wisdom that one should use the Lanczos method if no preconditioner is available. For most cases, JDQMR requires slightly fewer matvecs than JDCG, but JDCG is slightly faster because of the 15% more expensive iteration of QMR over CG. However, in three cases JDQMR is much faster than JDCG, justifying it as more robust, general purpose method. In the absence of a preconditioner, the matvec operation in sparse matrices is rarely dense enough to justify the more expensive step of GD+1. Yet, the matvecs in GD+1 are usually the closest (and often equal) to those of QMRopt. Therefore, a runtime check in the program can assess the expense of the matvec operation and if it is relatively high, as in matrices nd3kf and or56f, we choose GD+1 over JDQMR.

Preconditioners, when available, tend to dominate the expense at each step. In addition, as fewer iterations are needed with preconditioning, the effects of restarting are less dramatic. In such cases, we expect the GD+1 method to have an advantage.

Figure 5.4 shows the convergence history of various preconditioned methods on NASASRB.

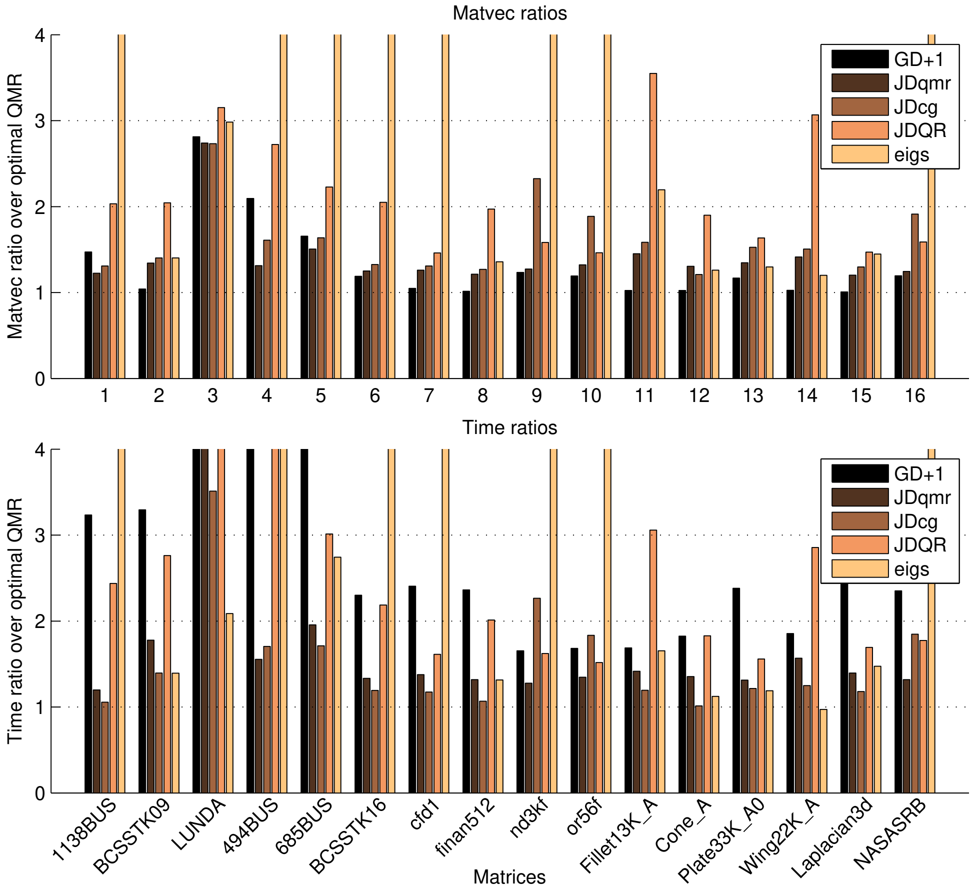

FIG. 5.3. *Ratios of matrix-vector operations (top graph) and time (bottom graph) taken by each of the five unpreconditioned methods over the the corresponding matvecs and time taken by the QMRopt. We show GD+1, JDQMR, JDCG, JDQR with 10 inner steps of QMR, and eigs(). LOBPCG was not competitive in these tests and is not shown. Matvecs for GD+1 are very close to QMRopt, followed closely by JDQMR. JDQMR is best time-wise, because of a less expensive inner step and sparse matrix vector products.*

TABLE 5.2

Number of matrix-vector operations performed for each test matrix by three unpreconditioned methods: the optimal QMRopt, GD+1, and ARPACK (eigs).

Matrix No.	1	2	3	4	5	6	7	8
Opt QMR	3468	359	362	1798	733	439	2437	318
GD+1	5105	374	1018	3767	1214	522	2558	323
eigs()	38376	504	1080	17032	3293	2142	9864	432

Matrix No.	9	10	11	12	13	14	15	16
Opt QMR	8518	4765	82	200	610	75	858	61448
GD+1	10506	5687	84	205	713	77	864	73474
eigs()	80910	30636	180	252	792	90	1242	-

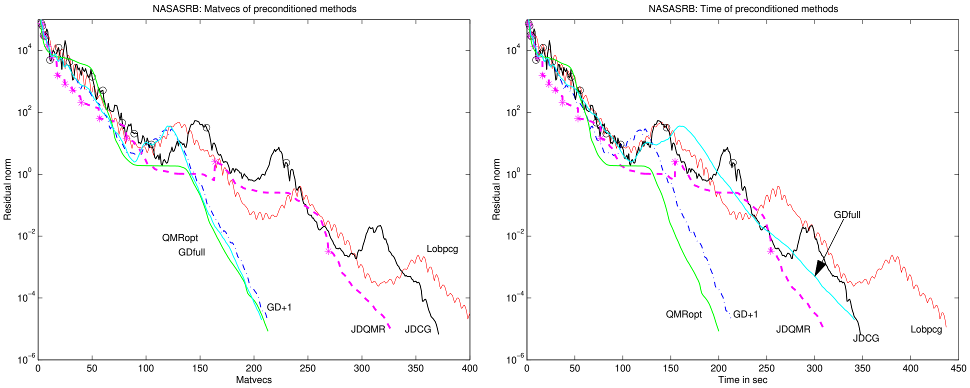

FIG. 5.4. *Residual convergence history for various preconditioned methods for matrix NASASRB. GD+1 converges identically to QMRopt and to the unrestarted GD. This shows the remarkable effectiveness of the +1 restarting scheme when coupled with subspace acceleration. LOBPCG is more than twice as slow. JDQMR and JDCG are also effective but not as much as GD+1. The inexpensive step of JDQMR/JDCG cannot outweigh the fewer applications of an expensive matrix-vector and preconditioning operators.*

TABLE 5.3

Number of matrix-vector (equivalently preconditioning) operations performed for each test matrix by three methods: the optimal QMRopt, the GD+1, and the SI-eigs using preconditioned QMR to solve for $(A-\eta I)^{-1}$ at every step. The top table shows those matrices that required at least one restart for GD+1. The performance of GD+1 is very close to optimal. When restarting is not needed (lower table) the full subspace optimization of GD+1 even improves on the QMR recurrence. Every step of the SI-eigs requires about as many matvecs as the optimal QMR method. Thus, even if only 2 steps were required, SI-eigs would not have been competitive with GD+1. Similar results are expected with inner-outer implementations of Inverse iteration or RQI.

Matrix No.	1	4	7	9	10	13	15	16
Opt QMR	49	29	218	1045	92	48	50	213
GD+1	53	31	237	1809	145	47	49	212
SI-eigs	458	247	2028	27683	1498	942	8553	2369

Matrix No.	2	3	5	6	8	11	12	14
Opt QMR	15	21	17	11	9	11	19	9
GD+1	13	17	16	7	8	12	18	9
SI-eigs	83	145	128	221	220	182	322	83

All methods perform within a factor of two of optimal, but GD+1 convergence is identical to optimal! Moreover, their relative convergence behaviors extend also to execution time, as the application of the `cholinc` preconditioner for this matrix dominates the costs. Based on the same principles, JDQMR and JDCG are similarly effective, with JDQMR slightly better.

Figure 5.5 shows a summary of results from applying five methods on our 16 matrices using preconditioning. We also provide the actual matvecs taken by QMRopt, GD+1, and SI-eigs in Table 5.3. Similarly to NASASRB, GD+1 is the clear winner in all cases, both iteration- and time-wise. Moreover, for all but two cases (nd3kf and or56f) its convergence is almost identical to optimal. Even for nd3kf and or56f, convergence is well within a factor of 2 from optimal. Clearly, inner-outer SI-eigs is far from competitive. Even adaptive inner-outer schemes for RQI and Inverse iteration as in [62, 6] cannot be any better than the subspace accelerated JDQMR.

Valuable intuition on the JDQMR convergence can be obtained through a closer look at its convergence curve in Figure 5.4. Recall that the CG/QMR convergence is usually de-

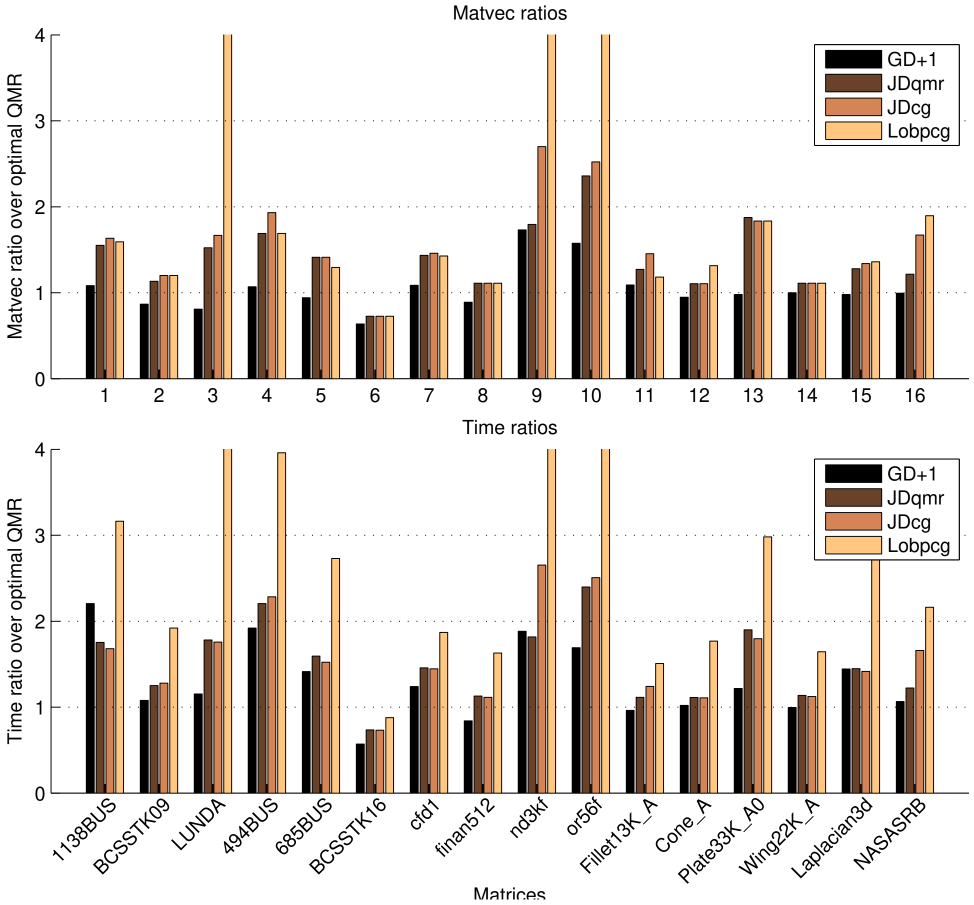

FIG. 5.5. *Ratios of matrix-vector operations (top graph) and time (bottom graph) taken by each of the four preconditioned methods over the the corresponding matvecs and time taken by the QMRopt. We show GD+1, JDQMR, JDCG, and LOBPCG. JDQR with 10 inner steps of QMR was not competitive in these examples and is not shown. Matvecs for GD+1 are impressively close to QMRopt and often lower, always improving on LOBPCG. JDQMR and JDCG are also very effective, but usually slower than GD+1.*

scribed by three phases; an initial linear, a plateau, and a final superlinear. At the beginning of the JDQMR curve, while the easiest components of the spectrum are being identified, stopping the inner QMR process frequently does not have any negative effects on the convergence. Later, the inner QMR builds a plateau from step 100 until it is stopped at step 160. This stopping causes the inner QMR to rebuild this plateau from step 180 to step 240. In Figure 5.4, this second plateau accounts primarily for the slow down over the optimal method. Fortunately, for the inner QMR to start to plateau, the outer JD must have reached the neighborhood of an eigenvalue. After that point, its convergence is cubic, similarly to RQI, and thus two or three outer steps are sufficient. A plateau is usually not expected during the third outer step, which is the one yielding the full accuracy. Therefore, we conjecture that JDQMR type methods cannot be more than three times slower than the optimal method, and usually they are significantly less than two times slower. All our experiments support this conjecture. Avoiding this repetition with truncated Newton methods (such as JDQMR) is an open problem that quasi-Newton type methods (such as GD+1) do not seem to have. Our criterion

TABLE 5.4
The effects of subspace acceleration on GD(m_{min}, m_{max})+1 without preconditioning. LOBPCG and GD(1,3)+1 are theoretically equivalent.

	1138BUS		494BUS		BCSSTK16		Laplacian3d		nd3kf	
	MV	sec	MV	sec	MV	sec	MV	sec	MV	sec
LOBPCG	7495	29.72	6919	20.00	584	14.33	1546	166.91	30275	1894.17
GD(1,3)+1	8242	30.43	6919	16.93	610	5.84	1752	54.71	30000	1401.02
GD(3,6)+1	4884	16.91	4158	11.13	503	5.48	943	36.64	15732	645.18
GD(5,8)+1	4904	19.36	4200	12.00	507	6.49	864	41.22	10860	470.21
GD(8,15)+1	5196	22.98	4604	14.19	486	7.34	864	51.24	9460	447.82
GD(15,30)+1	4967	30.50	3386	12.87	477	10.10	863	78.84	9060	539.91

TABLE 5.5
The effects of subspace acceleration on GD(m_{min}, m_{max})+1 with preconditioning.

	Plate33K_A0		cfd1		or56f		Laplacian3d		NASASRB	
	MV	sec	MV	sec	MV	sec	MV	sec	MV	sec
LOBPCG	88	23.23	311	244.44	>1001	>211.65	64	10.01	404	435.43
GD(1,3)+1	127	21.47	500	309.51	>1000	>166.10	64	5.40	403	379.51
GD(3,6)+1	54	9.77	289	182.35	195	33.57	51	4.64	224	215.42
GD(5,8)+1	50	9.31	251	166.09	161	28.01	49	4.95	213	207.71
GD(8,15)+1	47	9.52	231	163.41	144	25.14	49	5.42	212	213.19
GD(15,30)+1	46	11.15	230	179.51	109	19.58	48	6.35	212	227.52

(3.36) alleviates this problem.

5.3. The effects of subspace acceleration. We provide a set of experiments that shows the effects of increasing the basis size in the GD(m_{min}, m_{max})+1 method. In Table 5.4 we report the number of matvecs and running time for GD+1 with various basis sizes, and for LOBPCG on five matrices without preconditioning. First, we observe that GD(1,3)+1 and LOBPCG converge similarly, and often identically, supporting the theoretical equivalence of the two methods. The differences in timings are due to their different implementations and cache behaviors. Second, convergence improves drastically with a small subspace acceleration ($m_{max} = 6, 8$), but increasing m_{max} further only offers rapidly diminishing additional improvements. Consequently, the best time is achieved usually with $m_{max} = 6, 8$, unless the problem is too difficult and the matrix vector operator too expensive (see nd3kf case).

Similar results are reported in Table 5.5 using the same preconditioner as in section 5.2. Again even a small subspace acceleration on LOCG achieves a nearly optimal convergence. Note that the orthogonal basis of the GD(1,3)+1 may improve numerical stability in some problems (see next section), but not necessarily convergence (first two cases in Table 5.5). Finally, we note that the timings for GD(6,18)+1 in the previous section could improve if a smaller m_{max} of 6 or 8 were to be used.

5.4. Experiments with PRIMME. In Table 5.6 we provide comparisons between three state-of-the-art eigenvalue packages for symmetric eigenvalue problems. Our goal in developing PRIMME (PReconditioned Iterative MultiMethod Eigensolver) was to provide a robust and nearly optimal multimethod software for which users are not required to set any parameters. The adaptivity of the JDQMR and the relatively robust choices of parameters for GD+1 make these two methods excellent candidates for default eigensolvers. We compare against BLOPEX and JDBSYM. The first three matrices are preconditioned with ILUT [54] with its parameters chosen to provide a stable factorization. The last two matrices are unpreconditioned.

First, we observe that BLOPEX (LOPBCG), although it requires no parameter setting, is not competitive. In addition, in the cases where it failed to converge, it had reached a

TABLE 5.6

Comparison of three state-of-the-art preconditioned eigensolver codes. BLOPEX implements LOBPCG, JDB-SYM implements Jacobi-Davidson with sQMR as inner solver, and our PRIMME software includes both JDQMR and GD+1 which provide almost parameter-free near optimality.

	cfd1		or56f		Plate33K_A		cfd1		Cone_A	
	MV	sec	MV	sec	MV	sec	MV	sec	MV	sec
BLOPEX	669	114.14	332	21.89	-	-	6426	186.63	-	-
GD+1	270	49.92	174	11.56	272	39.89	2858	113.86	214	3.49
JDQMR	294	44.82	190	11.88	381	51.35	2370	49.45	281	3.19
JDBSYM	373	60.42	221	14.34	(747)	(102.6)	2412	48.95	(708)	(7.70)

residual norm of $\|A\|_F 10^{-10}$ but then encountered numerical problems. The JDBSYM was used with a symmetric QMR as inner solver, so the primary difference from JDQMR is the stopping criteria. The experiments confirm that when the JDBSYM criteria [13] capture the Newton convergence well, JDBSYM is close to JDQMR and sometimes competitive (unpreconditioned cfd1). The results in parentheses, however, show cases where JDBSYM could not converge, until the user provided a shift for the correction equation that was very close to the desired eigenvalue. GD+1 and JDQMR converge always, in the least time, and with no a priori information.

5.5. Interior eigenproblem with indefinite preconditioner. We borrow a model problem from *ab-initio* calculations that are common in many fields such as computational chemistry and materials science. In these applications, the Schrödinger operator is a sum of the Laplacian and certain local and non-local potential functions. We consider the eigenvalue problem stemming from the following simplified operator,

$$(-\nabla^2 + V)\psi = \varepsilon\psi, \tag{5.1}$$

where V is a local potential. In some cases, a few lowest eigenvalues are needed corresponding to the ground (most stable) states of the system. In many cases, however, we need a few eigenvalues around the Fermi energy level, a small energy gap that separates occupied from unoccupied states. Obtaining all eigenvalues lower than the Fermi level becomes extremely expensive for systems with large number of particles, suggesting that an interior eigenmethod should be preferable.

In real space discretizations V is a diagonal matrix. In planewave (Fourier) discretizations, however, ∇^2 becomes a diagonal matrix (and thus easily invertible) and V a dense matrix. Many planewave codes exploit this property, performing preconditioning with $(\nabla^2)^{-1}$ in Fourier space, while operating with V in real space.

Consider the problem 5.1 in the two dimensional unit square, with Neumann boundary conditions, and with a potential inversely proportional to the distance from z_0: $V(x) = 1/\|x - z_0\|$, with $z_0, x \in \Re^2$. We discretize the operator with a uniform finite difference, five point stencil, yielding the usual 5-diagonal Laplacian matrix with V added on the diagonal. We use $n = 110$ points in each direction, for a matrix dimension of 12100, and place the largest potential in the middle of the grid-square closest to 0: $z_0 = [0.5/n,\ 0.5/n]^T$. We look (arbitrarily) for the eigenvalue closest to 0.86, which is beyond the 800th lowest eigenvalue of our matrix. All computations are performed in real space, except preconditioning. To precondition a vector $\mathbf{r}$, we transform it through a fast sine transform to Fourier space, where the preconditioner

$$K^{-1} = (\nabla^2 - \eta_K I)^{-1} \tag{5.2}$$

is diagonal. An inverse sine transform returns the preconditioned vector to real space.

TABLE 5.7

The benefits of JDQMR for finding one interior eigenvalue closest to 0.86. *The lowest eigenvalue of A is greater than* -0.05. *In the upper table an interior problem is solved, where both preconditioner and correction equation can be indefinite. A positive definite preconditioner* ($\eta_K = -0.05$)*, as required by MINRES, does not converge. JDCG does not converge either when solving the correction equation for the extreme eigenvalue. The lower table shows that the transformation into an extreme eigenproblem for* $(A-0.86I)^2$ *is not a good alternative, even though JDQMR is far better than JDCG. ARPACK (eigs) without preconditioning cannot converge.*

Seek eigenvalue closest to $\sigma = 0.86$. ($\lambda_{exact} = 0.859768$), $-0.05 < \lambda_i < 7.99$				
Interior approach Correction equation with operator $(A-\eta I)$ and preconditioner $(\nabla^2 - \eta_K I)$				
Method	η	η_K	Matvecs	Equivalent method
QMRopt	λ_{exact}	λ_{exact}	261	Optimal correction/no dynamic stopping
GD+1	$\theta^{(m)}$	$\theta^{(m)}$	790	GD with recurrence restarting
JDQMR	$\theta^{(m)}$	$\theta^{(m)}$	1330	Adaptive accelerated RQI
JDQMR	0.86	$\theta^{(m)}$	1418	
JDQMR	0.86	0.86	1669	Adaptive accelerated INVIT
SI-eigs	0.86	0.86	12294	Shift-Invert ARPACK, QMR inner solver
JDQMR	0.86	-0.05	–	Adaptive JDMINRES
JDCG	-0.05	-0.05	–	
Exterior approach Smallest eigenvalue of $(A-0.86I)^2$ with preconditioner $(\nabla^2 - 0.86I)^2$				
Method			Matvecs	
GD+1			2642	
JDQMR			4490	
JDCG			12736	
eigs			$\gg 30000$	(no preconditioning)

Naturally, JDCG cannot be used if the shift η in the correction eq. (2.1) is inside the spectrum. Moreover, JD with MINRES as inner solver cannot be used either if η_K makes the preconditioner (5.2) indefinite. The question is whether treating the problem as an interior with indefinite preconditioner is beneficial. Table 5.7 shows the performance as the number of preconditioning operations (equivalently matrix vector products) for various methods. Because our Matlab implementation of the preconditioner is very expensive we do not report times.

GD+1 is the closest to the QMRopt method that uses the exact eigenvalue both in eq. (2.1) and in (5.2). Note that the threefold factor in slowdown does not reflect a decrease in convergence rate, but the fact that the solver spends most of the time trying to identify the proper eigenvalue to target. Once the eigenvalue is located, the convergence rate is similar to the optimal method. We have not investigated the use of harmonic Ritz values that could improve this identification. JDQMR converges in roughly 70% more iterations than GD+1. However, JDQMR too spends most of its time identifying the eigenpair, in fact exiting after one inner iteration. The astute reader may have noticed that in such cases the JDQMR is simply GD+1, only with two preconditioning operations per step. A more economical implementation is possible, but beyond our current discussion. As expected, fixing one or both of the shifts η, η_K makes convergence slightly worse, as demonstrated by various adaptive versions of INVIT. Without inner-outer adaptivity, the SI-eigs is an order of magnitude slower than JDQMR. For this example, JDMINRES with a positive definite preconditioner, or JDCG on a definite correction equation do not converge.

A common practice for interior eigenproblems is to solve for the smallest eigenvalue of $(A-\sigma I)^2$. This approach is attractively simple, but rarely competitive because of worse conditioning. The lower half of the Table 5.7 shows the performance of four methods on the squared matrix with a similarly squared preconditioner. The ratio between GD+1 and JDQMR iterations is the same as in the interior method, while the JDCG converges but significantly slower. As it cannot exploit preconditioning, eigs() cannot converge in tractable time.

6. Conclusions. Our goal is to develop a method, or multimethod, that converges near optimally and robustly for large, difficult eigenvalue problems. The method must operate under limited memory and be capable of using various preconditioners. We showed that such a method should approach the eigenvalue problem from the nonlinear perspective; either through the truncated Newton or through the limited memory quasi Newton techniques. Regardless of the technique, however, the underlying outer method must follow the GD outer scheme.

We described a truncated Newton technique, JDQMR, which extends the previously proposed JDCG method. JDQMR can work with indefinite correction equation and preconditioners, and thus can be used for finding interior eigenvalues, while providing better stopping criteria that improve robustness. In addition, we argued why JDQMR converges within a small factor (2 or 3) of optimal.

We also described methods that use the locally optimal CG either as a recurrence (LOBPCG) or as a restarting (GD+1), and unified them under the framework of limited memory quasi Newton. This framework, together with our previous theoretical results, provided new intuition on both the global and local convergence of these methods; for instance, explaining why GD+1 improves on unaccelerated recurrence methods such as LOBPCG.

A rather unexpected conclusion from our experiments is that limited memory BFGS type methods (GD+1) outperform truncated Newton methods (such as JDCG or JDQMR), in terms of convergence, even with a small acceleration basis. Equally impressive is the fact that GD+1 usually matches the convergence, both asymptotic and global, of the optimal QMRopt benchmark.

As the acceleration basis shrinks, the convergence of JDQMR, which is less sensitive to acceleration, matches or surpasses the convergence of GD+1 which, in turn, is always better than the extreme of using no acceleration basis (LOBPCG). Furthermore, when the preconditioner and matrix-vector operations are inexpensive, the cheaper inner iteration of JDQMR requires less overall time than GD+1. The advantage of our approach is that the GD driver implements both methods, switching at runtime when the actual expense of the user-provided operators is measured.

Finally, we have provided an extensive set of experiments. We considered a variety of problems and methods, with and without preconditioning, and problems that involved finding an interior eigenpair using indefinite preconditioning. Interestingly, both the theory and our experiments suggest that even without preconditioning, our multimethod should be preferred over ARPACK. Our results and conclusions extend also to finding a small number of eigenpairs. When many eigenvalues are required, the challenges and results are different; these topics are addressed in a companion paper [66].

Acknowledgement. The author would like to thank the two anonymous referees for their significant contribution to the clarity of the presentation, and for prompting the analysis in Section 3.4. The presentation also benefited from comments from the editor, Dr. T. Kolda, and the author's colleagues Profs. V. Torczon and P. Stockmeyer.

REFERENCES

[1] M. F. Adams. Evaluation of three unstructured multigrid methods on 3d finite element problems in solid mechanics. *International Journal for Numerical Methods in Engineerin*, 55:519–534, 2002.
[2] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq, and R. S. Tuminaro. A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods. *International Journal of Numerical Methods in Engineering*, 64:204–236, 2005.
[3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, 2000.
[4] 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.
[5] L. Bergamaschi, G. Gambolati, and G. Pini. Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem. *Numer. Lin. Alg. Appl.*, 4(2):69–84, 1997.
[6] J. Berns-Mueller. *Inexact Inverse Iteration using Galerkin Krylov solvers*. PhD thesis, University of Bath, UK, 2003.
[7] R. F. Boisvert, R. Pozo, K. Remington, R. Barrett, and J. Dongarra. The Matrix Market: A web resource for test matrix collections. In Ronald F. Boisvert, editor, *Quality of Numerical Software, Assessment and Enhancement*, pages 125–137, London, 1997. Chapman & Hall.
[8] W. W. Bradbury and R. Fletcher. New iterative methods for solution of the eigenproblem. *Numerische Mathematik*, 9:259–267, 1966.
[9] M. Clint and A. Jennings. The evaluation of eigenvalues and eigenvectors of a real symmetric matrix by simultaneous iteration. *Computer J.*, 13:76–80, 1970.
[10] M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.*, 15:62–76, 1994.
[11] 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.
[12] T. Davis. University of florida sparse matrix collection. Technical report, University of Florida. NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 19.
[13] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.
[14] E. G. D'yakonov. Iteration methods in eigenvalue problems. *Math. Notes*, 34:945–953, 1983.
[15] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1999.
[16] T. Eirola and O. Nevanlinna. Accelerating with rank-one updates. *Lin. Alg. and its Appl.*, 121:511–520, 1989.
[17] 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.
[18] R. W. Freund and N. M. Nachtigal. A new Krylov-subspace method for symmetric indefinite linear systems. Technical report, AT&T Bell Laboratories, Murray Hill, NJ, 1994.
[19] R. W. Freund and N. M. Nachtigal. Software for simplified Lanczos and QMR algorithms. *Applied Numerical Mathematics*, 19:319–341, 1995.
[20] G. Gambolati, F. Sartoretto, and P. Florian. An orthogonal accelerated deflation technique for large symmetric eigenproblems. *Comp. Methods App. Mech. Eng.*, 94:13–23, 1992.
[21] M. Genseberger and G. L. G. Sleijpen. Alternative correction equations in the Jacobi-Davidson method. *Numer. Linear Algebra Appl.*, (6):235–253, 1999.
[22] R. Geus. JDBSYM. http://people.web.psi.ch/geus/software.html.
[23] R. Geus. *The Jacobi-Davidson algorithm for solving large sparse symmetric eigenvalue problems with application to the design of accelerator cavities*. PhD thesis, ETH, 2002. Thesis. No. 14734.
[24] P. H. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1986.
[25] G. H. Golub and Q. Ye. Inexact inverse iteration for generalized eigenvalue problems. *BIT*, 40(4):671–684, 2000.
[26] G. H. Golub, Z. Zhang, and H. Zha. Large sparse symmetric eigenvalue problems with homogeneous linear constraints: The Lanczos process with inner-outer iterations. *Linear Algebra and its Applications*, 309:289–306, 2000.
[27] R. G. Grimes, J. G. Lewis, and H. D. Simon. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM J. Matrix Anal. Appl.*, 15(1):228–272, 1994.
[28] U. Hetmaniuk and R. B. Lehoucq. Basis selection in LOBPCG. *J. Comp. Phys.*, to appear, 2006.
[29] A. V. Knyazev. BLOPEX. http://www-math.cudenver.edu/ aknyazev/software/BLOPEX.
[30] A. V. Knyazev. Convergence rate estimates for iterative methods for mesh symmetric eigenvalue problem. *Soviet J. Numerical Analysis and Math. Modeling*, 2(5):371–396, 1987.
[31] A. V. Knyazev. Preconditioned eigensolvers - an oxymoron? *Electr. Trans. Numer. Anal.*, 7:104–123, 1998.
[32] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2001.
[33] A. V. Knyazev and K. Neymeyr. Efficient solution of symmetric eigenvalue problems using multigrid precon-

ditioners in the locally optimal block conjugate gradient method. *ETNA*, 15:38–55, 2003.

[34] A. V. Knyazev and K. Neymeyr. A geometric theory for preconditioned inverse iteration. iii: A short and sharp convergence estimate for generalized eigenvalue problems. *Linear Algebra Appl.*, 358:95–114, 2003.

[35] T. G. Kolda, D. P. O'Leary, and L. Nazareth. Bfgs with update skipping and varying memory. *SIAM Journal on Optimization*, 8(4):1060–1083, 1998.

[36] Y.-L. Lai, K.-Y. Lin, and W.-W. Lin. An inexact inverse iteration for large sparse eigenvalue problems. *Num. Lin. Alg. Appl.*, 4:425–437, 1997.

[37] R. B. Lehoucq and K. Meerbergen. Using generalized cayley transformations within an inexact rational Krylov sequence method. *SIAM J. Matrix Anal. Appl.*, 20(1):131–148, 1999.

[38] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, PA, 1998.

[39] B. Liu. Numerical algorithms in chemistry: Algebraic methods, eds. c. moler and i. shavitt. Technical Report LBL-8158, Lawrence Berkeley Laboratory, 1978.

[40] E. Lundström and L. Eldén. Adaptive eigenvalue computations using Newton's method on the Grassmann manifold. *SIAM Journal on Matrix Analysis and Applications*, 23(3):819–839, July 2002.

[41] K. Meerbergen. *Robust methods for the calculation of rightmost eigenvalues of nonsymmetric problems*. PhD thesis, Department of Computer Science, K. U. Leuven, Heverlee, Belgium, 1996.

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

[43] 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.

[44] 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.

[45] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer-Verlag, New York, 1999.

[46] Y. Notay. Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem. *Numerical Linear Algebra with Applications*, 9:21–44, 2002.

[47] Y. Notay. Is Jacobi-Davidson faster than Davidson? *SIAM J. Matrix Anal. Appl.*, 26(2):533–543, 2005.

[48] J. Olsen, P. Jørgensen, and J. Simons. Passing the one-billion limit in full configuration-interaction (FCI) calculations. *Chem. Phys. Lett.*, 169(6):463–472, 1990.

[49] E. Ovtchinnikov. Convergence estimates for the generalized Davidson method for symmetric eigenvalue problems I: The preconditioning aspect. *SIAM Journal on Numerical Analysis*, 41(1):258–271, February 2004.

[50] E. Ovtchinnikov. Convergence estimates for the generalized Davidson method for symmetric eigenvalue problems II: The subspace acceleration. *SIAM Journal on Numerical Analysis*, 41(1):272–286, February 2004.

[51] C. C. Paige and D. M. Saunders. solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–624, 1975.

[52] B. N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM, Philadelphia, PA, 1998.

[53] A. Ruhe and T. Wiberg. The method of conjugate gradients used in inverse iteration. *BIT*, 12(4):543–554, 1972.

[54] Y. 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/>.

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

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

[57] A. Sameh and Z. Tong. The trace minimization method for the symmetric generalized eigenvalue problem. *J. Comput. Appl. Math.*, 123:155–175, 2000.

[58] V. Simoncini and L. Eldén. Inexact Rayleigh quotient-type methods for eigenvalue computations. *BIT Numerical Mathematics*, 42(1):159–182, 2002.

[59] 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.

[60] 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.

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

[62] P. Smit and M. H. C. Paardekooper. The effects of inexact solvers in algorithms for symmetric eigenvalue problems. *Linear Algebra Appl.*, 287(1-3):337–357, 1999. Special issue celebrating the 60th birthday of Ludwig Elsner.

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

[64] 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.

[65] A. Stathopoulos and C. F. Fischer. A Davidson program for finding a few selected extreme eigenpairs of a large, sparse, real, symmetric matrix. *Computer Physics Communications*, 79(2):268–290, 1994.

[66] A. Stathopoulos and J. R. McCombs. Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part II: Seeking many eigenvalues. *Submitted*, (Technical Report WM-CS-2006-02).

[67] A. Stathopoulos, Serdar Öğüt, Y. Saad, J. R. Chelikowsky, and Hanchul Kim. Parallel methods and tools for predicting material properties. *Computing in Science and Engineering*, 2(4):19–32, 2000.

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

[69] A. Stathopoulos, Y. Saad, and C. F. Fischer. Robust preconditioning of large, sparse, symmetric eigenvalue problems. *Journal of Computational and Applied Mathematics*, 64:197–215, 1995.

[70] 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.

[71] D. Szyld. Criteria for combining inverse iteration and Rayleigh quotient iteration. *SIAM J. Num. Anal.*, 25(6):1369–1375, 1988.

[72] H. A. van der Vorst. *Iterative Krylov methods for large linear systems*, volume 13 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, UK, 2003.

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

[74] K. Wu and H. D. Simon. Thick-restart Lanczos method for symmetric eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 22(2):602–616, 2001.

[75] C. Yang, B. W. Peyton, D. W. Noid, B. G. Sumpter, and R. E. Tuzun. Large-scale normal coordinate analysis for molecular structures. *SIAM Journal on Scientific Computing*, 23(2):563–582, 2001.