Dynamic Thick Restarting of the Davidson, and the Implicitly Restarted Arnoldi Methods *

Andreas Stathopoulos, Yousef Saad, Kesheng Wu
Computer Science Department, University of Minnesota

October 29, 1996

Abstract

The Davidson method is a popular preconditioned variant of the Arnoldi method for solving large eigenvalue problems. For theoretical, as well as practical reasons the two methods are often used with restarting. Frequently, information is saved through approximated eigenvectors to compensate for the convergence impairment caused by restarting. We call this scheme of retaining more eigenvectors than needed 'thick restarting', and prove that thick restarted, non-preconditioned Davidson is equivalent to the implicitly restarted Arnoldi. We also establish a relation between thick restarted Davidson, and a Davidson method applied on a deflated system. The theory is used to address the question of which and how many eigenvectors to retain and motivates the development of a dynamic thick restarting scheme for the symmetric case, which can be used in both Davidson and implicit restarted Arnoldi. Several experiments demonstrate the efficiency and robustness of the scheme.

Key words. Davidson method, Arnoldi method, Lanczos method, implicit restarting, deflation, eigenvalue, preconditioning

AMS Subject Classification. 65F15

1 Introduction

The computation of a few eigenpairs of large, sparse, eigenvalue problems $Ax = \lambda x$, is central to many scientific applications [19]. The Arnoldi method, and its equivalent in the symmetric case the Lanczos method, have been the traditional approach to solving these problems. Preconditioning, through some shift-and-invert technique [22], is frequently employed to improve robustness. A different approach is followed by the Generalized Davidson (GD) method [8, 16, 6] which is a popular preconditioned variant of the Lanczos iteration. Instead of using a three-term recurrence to build an orthonormal basis for the Krylov subspace, the GD algorithm obtains the next basis vector by explicitly orthogonalizing the preconditioned residual $(M - \lambda I)^{-1}(A - \lambda I)x$ against the existing basis. A straightforward extension to the nonsymmetric case has also been studied in [21]. When $M = A$, the preconditioned residual yields back x, thus providing no improvement. The Jacobi-Davidson (JD) modification, proposed in [23], suggests that the proper way to precondition the residual is through an operator with range orthogonal to x. The GD

*Work supported by NSF grants DMR–9217287 and ASC 95-04038, and the Minnesota Supercomputer Institute

and its JD modification can be regarded as two ways of improving convergence and robustness at the expense of a more complicated step.

Often, eigenvalue problems are very large and ill-conditioned. As a result, eigenvalue methods require a large number of steps, and need to save all the vector iterates for extracting the eigenvectors. Such cases exhibit overwhelming storage requirements. In addition, the Lanczos and Arnoldi processes, which traditionally had been considered without restarting, are plagued by orthogonality problems and spurious solutions. For the above reasons many restarting variants are used in practice [7, 18, 24, 2]. The GD method improves convergence, and solves some of the aforementioned problems through orthogonalization and preconditioning. However, the number of iterations can still grow very large, and cause similar storage problems. The problem is actually aggravated in the symmetric case, where the better theoretical framework and software has led researchers to consider matrices of huge size that allow only a few vectors to be stored. The GD method also can be restarted every time the basis contains m vectors (GD(m)). If the l lowest eigenvalues are needed, the l lowest Ritz values are computed at the m_{th} step, and their corresponding Ritz vectors are used as initial guesses for the restarted GD iteration.

Truncating the Krylov sequence is expected to impair the convergence rate of the method. There are two main reasons: the new vectors entering the basis repeat some of the information that was discarded when restarting, and the Rayleigh-Ritz procedure does not minimize over the whole Krylov subspace. There has been much discussion about the problems caused by restarted methods for both linear systems and eigenvalue problems [27, 20, 24]. Some methods tend to save additional information at each restart [14, 3, 11]. For the Davidson method, Murray et al. [17], and Van Lenthe et al. [29], have proposed restarting with two vectors per required Ritz vector with some success. In an effort to minimize execution time, Crouzeix et al. [6], have proposed a dynamically chosen size m.

Recently, 'implicit restarting' has gained popularity as a means of improving convergence of the restarted Arnoldi procedure [24]. By using $p = m - k$ steps of the implicit QR algorithm on the Hessenberg matrix, the basis is truncated down to k vectors. It turns out that the k new basis vectors can be considered the Arnoldi vectors obtained from a polynomially transformed starting vector. This is the basis of the popular eigenvalue package ARPACK [13]. Preconditioners for eigenvalue problems usually vary between steps, in which case the Implicitly Restarted Arnoldi (IRA(k, m)) is not straightforward to apply. Further, in case of the GD(m) where the residual is preconditioned, the Ritz vectors can not be described with a polynomial of A. Clearly, a new restarting scheme is needed.

In this paper, we study an extension to the IRA(k, m) technique for the GD(m), which we call 'thick restarting' and denote by GD(k, m), and which depends on an integer parameter k. GD(k, m) restarts with k Ritz vectors instead of the l wanted ones, where $l \leq k < m$. The principle idea is mentioned by Kosugi in [11], Sleijpen et al. in [23] and Morgan in [15]. In the literature, the benefits of IRA(k, m) are studied in relation to the polynomially transformed initial vector. This paper addresses the question of which and how many Ritz vectors should be kept. The theory presented motivates a dynamic strategy of thick restarting, that can be used in both IRA(k, m) and GD(k, m). Although the results are proved for the non-preconditioned case, the idea of thick restarting is readily applicable to the preconditioned GD(k, m) and similar behavior is expected. Compared with IRA(k, m), GD(k, m) can also assume any number of initial guesses, and/or enhancements of the basis through arbitrary vectors, during the procedure.

After briefly presenting the IRA(k, m) and GD(k, m) algorithms in section 2, in section 3 we prove as an extension to [15] that in the absence of preconditioning, and for arbitrary targeting scheme of GD(k,m), the IRA(k,m) using the Ritz-values as shifts, and GD(k,m) are equivalent, in the sense that their basis vectors span exactly the same space. In section 4 a theorem is proved that relates the IRA(k,m), and thus GD(k,m), with an Arnoldi process applied on an approximately deflated initial vector. This extends the ideas that appeared recently in [28]. In section 5, a dynamic choice of k is derived for the symmetric case, where the rate of convergence is described by well-known bounds. In section 6, numerical experiments on matrices from the Harwell Boeing collection demonstrate the effectiveness of GD(k,m).

2 The restarted Arnoldi and Davidson methods

Throughout this paper we assume that the matrix A is diagonalizable, of order N, with eigenpairs (λ_i, x_i). We look for l outermost eigenpairs (e.g., lowest or highest in the symmetric case). The Arnoldi and Davidson methods use a basis size of $m > l$. The following descriptions of the algorithms serve for establishing the notation. For theoretical and implementation details refer to [24, 13, 8, 16, 6, 23]. For all quantities, the superscripts in parentheses denote the corresponding restarting step. These superscripts are dropped whenever there is no ambiguity.

Restarted Arnoldi's method in its simplest form can be expressed as follows:

Algorithm **2.1 Restarted Arnoldi**

0. Start: Choose initial unit vector $v^{(0)}$
1. For $s = 0, 1, \ldots$ *Do*
2. $\quad v_1 = v^{(s)}$, $V_1^{(s)} = \{v_1\}$
3. $\quad$ *For* $j = 1, \ldots, m$ *Do*
4. $\quad\quad h_{ij} = (Av_i, v_j),\ i = 1, \ldots, j,$
5. $\quad\quad w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
6. $\quad\quad h_{j+1,j} = \|w_j\|_2$, *if* $h_{j+1,j} = 0$ *stop.*
7. $\quad\quad v_{j+1} = w_j / h_{j+1,j}$
8. $\quad$ *Enddo*
9. $\quad$ *Compute the wanted eigenpairs* $(\mu_i^{(s)}, y_i^{(s)})$ *of* $H_m^{(s)} = (h_{i,j})$
$\quad\quad$ *and the Ritz vectors* $x_i^{(s)} = V_m^{(s)} y_i^{(s)}$, *where* $V_m^{(s)} = \{v_1, \ldots, v_m\}$
10. $\quad v^{(s+1)} = \sum c_i x_i^{(s)}$, *for some* c_i, *and the wanted* $x_i^{(s)}$
11. Enddo

The algorithm builds a Hessenberg matrix, from which the approximate eigenpairs are extracted through the Rayleigh-Ritz procedure. For the symmetric case, $H_m^{(s)}$ is a tridiagonal matrix, and a three-term recurrence replaces the above orthogonalization step. A linear combination of the wanted Ritz vectors are used to restart the algorithm. Such a restarting strategy however, may discard a lot of information and result in degradation of the convergence rate.

Implicitly restarted Arnoldi applies the implicit QR algorithm with the $m - l$ unwanted eigenvalues as shifts to the Hessenberg matrix, and uses the generated orthogonal transformations to truncate the basis down to l vectors. Therefore, it avoids the need to restart with a single vector which captures the information for all l eigenvectors. The number of vectors in the new basis after restart, may also be larger than l, say k. For the rest of the paper we assume that $l \leq k < m$, $p = m - k$, and IRA(k,m) denotes the associated method. An outline of the IRA(k,m) algorithm follows:

Algorithm **2.2 Implicitly Restarted Arnoldi**

0. *Start: Choose initial residual vector* $v_1^{(0)}$
1. *Build an initial Arnoldi iteration of* k *steps:* $(V_k^{(0)}, H_k^{(0)})$
2. *For* $s = 0, 1, \ldots$ *Do*
3. *Test for convergence*
4. *Extend* $V_k^{(s)}$ *to* $k+p$ *vectors, taking* p *more Arnoldi steps:* $(V_{k+p}^{(s)}, H_{k+p}^{(s)})$
5. *Choose shifts* $\mu_i,\ i = 1, \ldots, p$
6. $H_{k+p} = Q^T H_{k+p}^{(s)} Q$, *with* Q *the orthogonal matrix obtained through the implicit QR algorithm with* $\mu_i, i = 1, \ldots, p$ *shifts*
7. *Define* $V_k^{(s+1)} = (V_{k+p}^{(s)} Q) \begin{pmatrix} I_k \\ 0 \end{pmatrix}$, *and*
8. $H_k^{(s+1)} = \begin{pmatrix} I_k & 0 \end{pmatrix} H_{k+p} \begin{pmatrix} I_k \\ 0 \end{pmatrix}$
9. *Enddo*

The power of the IRA(k, m) lies in the following two properties. First,

$$v_1^{(s+1)} = \psi(A) v_1^{(s)} = \prod_{i=1}^{p} (A - \mu_i I) v_1^{(s)}, \tag{1}$$

for any choice of shifts μ_i, not limited to the exact shifts (Ritz values), and thus the new Arnoldi iteration starts with a polynomially transformed initial vector. Second, the vectors $v_2^{(s+1)}, \ldots, v_k^{(s+1)}$ can be considered the Arnoldi vectors of the Arnoldi process started with $v_1^{(s+1)}$. Thus, no matrix vector multiplications are needed for the first k Arnoldi vectors. Among various interpretations, IRA(k, m) can be considered a truncation of the QR algorithm for dense matrices, as well as an efficient and robust implementation of the subspace iteration with polynomial transformations.

The Davidson method first appeared as a diagonally preconditioned version of the Lanczos method for the symmetric eigenproblem. Extensions, to both general preconditioners and to the nonsymmetric case have been given since. The following describes the algorithm for the symmetric case. For the nonsymmetric case, line 5 should also include the computation of the last row of the projection matrix $T_j^{(s)}$.

Algorithm **2.3 Generalized Davidson**

0. *Choose initial unit vectors* $U_l^{(0)} = \{u_1^{(0)}, \ldots, u_l^{(0)}\}$
1. *For* $s = 0, 1, \ldots$ *Do*
2. $w_i^{(s)} = A u_i^{(s)},\ i = 1, \ldots, l-1$
3. *For* $j = l, \ldots, m$ *Do*
4. $w_j^{(s)} = A u_j^{(s)}$.
5. $t_{i,j} = (w_j^{(s)}, u_i^{(s)}),\ i = 1, \ldots, j$, *the last column of* $T_j^{(s)}$
6. *Compute some wanted eigenpair, say* (μ_1, z_1) *of* $T_j^{(s)}$.
7. $x_1 = U_j^{(s)} z_1$ *and* $r = A x_1 - \mu_1 x_1$, *the Ritz vector and its residual*
8. *Test* $\|r\|$ *for convergence. If satisfied target a new vector.*
9. *Solve* $M_{(s,j)} t = r$, *for* t.

10. $\quad b_{j+1}^{(s)} = MGS(U_j^{(s)}, t)$
11. $\quad$ *Enddo*
12. $\quad$ *Set* $U_k^{(s+1)} = \{x_1, \ldots, x_k\},\ k < m,$ *and restart*
13. Enddo

The preconditioning is performed by solving the equation at step 9. In [23] Sleijpen et al. show that for stability, robustness, as well as efficiency, the operator $M_{(s,j)}$ should have a range orthogonal to x. The method is called Jacobi-Davidson (JD), and it solves approximately the projected correction equation:

$$(I - x_1 x_1^T)(A - \mu_1 I)(I - x_1 x_1^T)\ t = (I - x_1 x_1^T)(\mu_1 I - A) x_1.$$

The projections can be easily applied if an iterative linear solver is used. For preconditioners which approximate A directly, such as incomplete factorizations and approximate inverses, the above orthogonality condition is enforced through an equivalent formulation known as Olsen method. Since the purpose of this paper is the study of restarting strategies, we use the general description of GD, and the results are valid whether step 9 is performed through JD or otherwise.

A Davidson step is more expensive than that of the Lanczos and Arnoldi algorithms, to allow for preconditioning. In addition, the Davidson algorithm can start with any number of initial vectors, and include in the basis any extra information that can be available during the execution. The targeted eigenpair (i.e., the one chosen for preconditioning) may vary in different steps, allowing for a variable targeting scheme. Finally, it can restart with the approximate eigenvectors, so it does not share the problems of the original Restarted Arnoldi. As in IRA(k,m), the Davidson method can also restart with more Ritz vectors than needed. We call this version 'thick restarting' and denote by GD(k,m), where l, k, and m are defined as in IRA(k,m). In the following section, we show that IRA(k,m) and GD(k,m) are equivalent in the non-preconditioned case, but GD(k,m) offers all the aforementioned advantages and extensions.

3 Thick and implicit restarting

It is known that the Lanczos and the Davidson methods are equivalent when no preconditioning is used. However, this has been pointed out only for the non-restarted case, where one eigenvalue is sought [16]. Recently, the equivalence of the IRA(k,m) with an Arnoldi method restarting with a Ritz vector and augmented by $k-1$ Ritz vectors has been shown [15]. In this section we prove that in the non-preconditioned case, if GD(k,m) starts with one initial vector, IRA(k,m) and GD(k,m) are equivalent, for any targeting scheme of GD(k,m).

The first Lemma is an extension of Lemma 3.10 in [24], and it is the basis for the equivalence proof. Note that the implicit QR algorithm is applied to any diagonalizable matrix H.

Lemma 3.1 *Let* $\lambda(H) = \{\lambda_1, \ldots, \lambda_k\} \cup \{\mu_1, \ldots, \mu_p\}$ *be a disjoint partition of the eigenvalue set of a diagonalizable matrix* H. *Let* $Q = Q_1 Q_2 \cdots Q_p$, *where* Q_i *is the orthogonal matrix implicitly defined by the shift* μ_i *in the implicit QR algorithm on* H. *Then, the first* k *columns of* Q *span the same space as the* k *eigenvectors* y_i *of* H *associated with the eigenvalues* $\lambda_i,\ i = 1, \ldots, k$.

Proof. After p steps of the implicit QR algorithm, it holds:

$$QR = Q_1 Q_2 \cdots Q_p R_p \cdots R_2 R_1 = \prod_{i=1}^{p} (H - \mu_i),$$

where Q_iR_i is the QR decomposition of $H_i - \mu_i$ at the i_{th} step, and $R = (r_{ij}) = R_p \cdots R_2R_1$ denotes an upper triangular matrix. Since the shifts $\mu_i,\ i = 1,\dots,p$ are eigenvalues of H, QR is a rank k matrix, and if the decompositions are performed with traditional column pivoting, $r_{ii} \neq 0,\ i = 1,\dots,k$ and $r_{ii} = 0,\ i = k+1,\dots,k+p$. For Hessenberg matrices it is shown in [24] that $q_1 = Qe_1$ is in the span of $\{y_1,\dots,y_k\}$. Using a similar argument, if $e_1 = \sum_{j=1}^{k+p} \xi_j y_j$,

$$QRe_1 = q_1 r_{11} = \sum_{j=1}^{k} \xi_j \prod_{i=1}^{p} (\lambda_j - \mu_i) y_j,$$

and $q_1 \in \text{span}\{y_1,\dots,y_k\}$. Inductively, let $q_1,\dots,q_s \in \text{span}\{y_1,\dots,y_k\}$. If $e_{s+1} = \sum_{j=1}^{k+p} \xi_{s,j} y_j$,

$$QRe_{s+1} = \sum_{j=1}^{s} r_{j,s+1} q_j + r_{s+1,s+1} q_{s+1} = \sum_{j=1}^{k} \xi_{s,j} \prod_{i=1}^{p} (\lambda_j - \mu_i) y_j,$$

and since $r_{s+1,s+1} \neq 0,\ q_{s+1} \in \text{span}\{y_1,\dots,y_k\}$. Since Q is orthogonal matrix, the first k columns of Q are independent and therefore $\text{span}\{q_1,\dots,q_k\} = \text{span}\{y_1,\dots,y_k\}$. □

In the special case where the matrix H is the Hessenberg matrix obtained from the Arnoldi procedure, an immediate consequence is the following:

Lemma 3.2 *If at step s the basis vectors $U_m^{(s)}$ and $V_m^{(s)}$ of $GD(k,m)$ and $IRA(k,m)$ respectively, span the same space, then, after restarting both methods,*

$$\text{span}(V_k^{(s+1)}) = \text{span}(U_k^{(s+1)}).$$

Proof. From the assumption, the Ritz vectors are the same for both methods at the end of the s-th step, and after restarting, $U_k^{(s+1)}$ contains the k chosen ones, say X_k. If $Q(1:k)$ are the first k columns of the orthogonal matrix of Lemma 3.1, we have: $V_k^{(s+1)} = V_m^{(s)} Q(1:k) = V_m^{(s)} Y(1:k) C = X_k C$, where $Y(1:k)$ are the chosen k eigenvectors of the Hessenberg matrix H_m, and C some $k \times k$ coefficient matrix. □

The above shows the equivalence of the two methods at restart. To conclude the proof we need the following proposition which describes the residuals of the Ritz vectors of the Arnoldi procedure [19].

Proposition 3.1 *At the j_{th} step of inner Arnoldi loop, let y_i be the i_{th} eigenvector of H_j associated with the eigenvalue λ_i, and x_i the Ritz approximate eigenvector $x_i = V_j y_i$. Then,*

$$(A - \lambda_i I) x_i = h_{j+1,j} e_j^H y_i \ v_{j+1}.$$

Theorem 3.1 *If $GD(k,m)$ without preconditioning and $IRA(k,m)$ are executed with the same initial vector $v^{(0)}$, and at each restarting the p shifts used in $IRA(k,m)$ are the Ritz values of the Ritz vectors discarded by $GD(k,m)$, then the basis vectors produced by the two methods span the same space, for any targeting scheme of $GD(k,m)$, and thus the methods are equivalent.*

Proof. If the two methods start with the same initial vector and no restarting is used, the vectors built are identical. This is an immediate consequence of Proposition 3.1, for any selection of targets in GD(k,m). This is well established in the literature (see [16, 21]).

For the general case, a simple induction on the number s of restarts is used. From the above, it follows that for $s=0$, the bases built by IRA(k,m) and GD(k,m) satisfy $V_m^{(0)} = U_m^{(0)}$.

Let for $s>0$, $\mathrm{span}(V_m^{(s)}) = \mathrm{span}(U_m^{(s)})$. After restarting both methods, and from Lemma 3.2, $\mathrm{span}(V_k^{(s+1)}) = \mathrm{span}(U_k^{(s+1)})$. As a result, at this k step, the Ritz vectors for both methods are the same, and because of Proposition 3.1, the next expansion vectors for both methods are parallel. Thus it holds, $\mathrm{span}(V_{k+1}^{(s+1)}) = \mathrm{span}(U_{k+1}^{(s+1)})$, and inductively,

$$\mathrm{span}(V_m^{(s+1)}) = \mathrm{span}(U_m^{(s+1)}).$$

$\square$

A few comments are in order. Lemma 3.1 can be applied to the Hessenberg matrices built by Krylov subspace methods, if these are diagonalizable. This assumption is always satisfied by the tridiagonal matrices built in the symmetric case. This justifies the use of this result in Lemma 3.2, for the non-preconditioned case.

Further, Lemma 3.1 applies to any non-Hessenberg diagonalizable matrix, and although Lemma 3.2 discusses the IRA(k,m) method, it is true for all methods that use an implicit restarting scheme. Consequently, implicit restarting can be applied to the projection, full matrix T obtained from the preconditioned basis vectors of GD(k,m). If exact shifts are used, it produces a sequence of vectors that span the same space with the required Ritz vectors. Several numerical examples, however, have shown that this can be an unstable process. The reason is traced back to the forward numerical instability of the QR process. Treatments of the problem have been developed [12], but we find it inexpensive and stable to thick restart with the orthogonal (or orthogonalized in the nonsymmetric case) Ritz vectors.

In the preconditioned case, the application of implicit restarting does not result in a polynomial transformation as in equation (1). Specifically, let $U = \{u_1^{(0)}, \ldots, u_m^{(0)}\}$ be the GD(k,m) basis before restarting, with decomposition $AU = UT + E$ and $U^H E = 0$. Following similar steps to Lemma 3.1 and to those in [24], the first basis vector $u_1^{(1)}$ after the implicit restarting can be expressed as: $u_1^{(1)} = \psi(UU^H AUU^H)u_1^{(0)} = U\psi(T)U^H u_1^{(0)}$, where ψ is an appropriate polynomial. The polynomial transformation involves the projected matrix on the space spanned by U, and not the full rank matrix A. An arbitrary choice of shifts may lead to a different polynomial but there are no clear advantages for doing so.

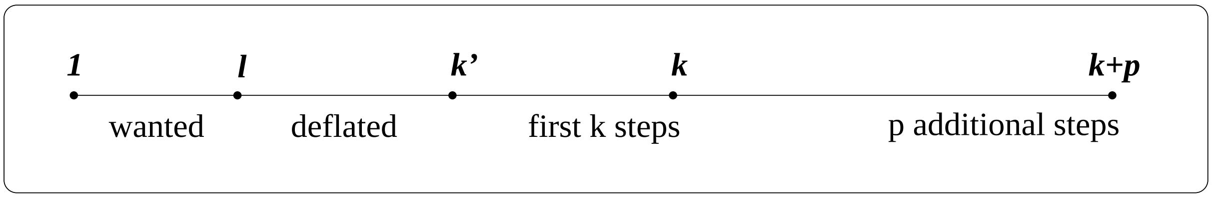
Finally, in the unlikely case where preconditioning produces a defective projection matrix, both implicit and thick restarting may fail as described earlier. Working with the Schur vectors, rather than the Ritz vectors provides a stable solution to the problem. The algorithm has been proposed recently in [10] and consists of a slight modification to the GD(k,m): Instead of finding the eigendecomposition of T, a Schur decomposition is computed and the diagonal elements of the upper triangular matrix are used as shifts in the implicit restarting procedure. In this way the algorithm computes a partial Schur decomposition of A. Note that thick restarting can still be applied keeping more Schur vectors than needed.

4 The deflation connection

Krylov methods for linear systems, such as conjugate gradient (CG) and GMRES, demonstrate a superlinear convergence at later iterations. One explanation of this phenomenon is the convergence of the outermost eigenpairs of the matrix, so that each method behaves as if deflation has occurred, resulting in faster convergence. Such observations have appeared as early as in [5], but actual quantification of the behavior appears in [20] and [27, 28]. In the latter papers, the optimality of the CG and GMRES polynomials is employed to relate each method after some iterations with a similar process of the same method on a deflated residual.

Results similar to [28] can not be applied directly to the residual and eigenvalues in the nonsymmetric Arnoldi, since there is no optimality principle. In the following, we extend the results found in [28] to the Arnoldi method, by considering the distance of some eigenvector from the Arnoldi Krylov subspace. Again preconditioning is not considered since the space that it creates is not a Krylov subspace. This general result is used later in the context of thick/implicit restarting to justify the expected benefits and to help provide a good choice of k.

For simplicity, let A be a diagonalizable matrix, $X^{-1}AX = \Lambda = \mathrm{diag}(\lambda_i)$, of order N. The results in this section can be extended naturally to the Jordan form of A, following the methodology in [28]. However, the presentation is more involved. Let $v = X\xi$ be the expansion of the starting Arnoldi vector to the eigenvector basis. Define three numbers satisfying $l < k' \leq k$, where $k-1$ is the number of steps that a non-restarted Arnoldi method takes starting from v. We can assume an eigenvalue ordering so that the first l ones are wanted, and the eigenvalues $l+1, \ldots, k'$ are well approximated by the $k-1$ steps of Arnoldi. Let μ_i be the k Ritz values from this $\mathcal{K}_k(v)$ space. At this point we let the Arnoldi process take p more steps and build the space $\mathcal{K}_{k+p}(v)$. The following shows the ordering of these numbers:



Define $\mathcal{D}^{(k)}$ a diagonal matrix with elements:

$$\mathcal{D}_{jj}^{(k)} = \begin{cases} 0, & \text{for } j = l+1, \ldots, k' \\ \prod_{i=l+1}^{k'} \frac{\lambda_j - \lambda_i}{\lambda_j - \mu_i}, & \text{for } j \leq l \text{ or } j = k'+1, \ldots, N \end{cases} . \tag{2}$$

Assuming the above definitions we have the following theorem:

Theorem 4.1 *Let x_j an eigenvector to be approximated from the Krylov subspace $\mathcal{K}_{k+p}(v)$, and $\tilde{x}_j$ the corresponding Ritz vector from $\mathcal{K}_k(v)$, whose components of $x_{l+1}, \ldots, x_{k'}$ have been removed. If these Krylov subspaces can be built, then for any $j = 1, \ldots, l$:*

$$\mathrm{dist}(x_j, \mathcal{K}_{k+p}(v)) \leq |1 - \mathcal{D}_{jj}^{(k)}| + \|X\mathcal{D}^{(k)}X^{-1}\| \ \mathrm{dist}(x_j, \mathcal{K}_p(\tilde{x}_j)).$$

Proof. At step $k-1$ of the Arnoldi procedure, the Ritz vector x_j' from $\mathcal{K}_k(v)$ has the following expression:

$$x_j' = q_j(A)v / \|q_j(A)v\|, \quad \text{with}$$

$$q_j(t) = \prod_{i=1, i\neq j}^{k} (t - \mu_i).$$

We define $h(t)$ a polynomial of degree $k-1$ as:

$$h(t) = \prod_{i=l+1}^{k'} \frac{(t-\lambda_i)}{(t-\mu_i)} q_j(t).$$

Note that the eigen-components $l+1, \ldots, k'$ of the vector $h(A)v$ are annihilated. If $\tilde{\xi}_i = 0$ for $i = l+1, \ldots, k'$ and $\tilde{\xi}_i = \xi_i$ otherwise, then $\tilde{x}_j = \frac{1}{\phi} X q_j(\Lambda)\tilde{\xi}$, where ϕ is a normalization factor. Since any vector in $\mathcal{K}_{k+p}(v)$ can be expressed as a polynomial of A applied on v, if π^* is some polynomial of degree p, and e_j the j_{th} orthocanonical vector, we have:

$$\begin{aligned}
\mathrm{dist}(x_j, \mathcal{K}_{k+p}(v)) &= \min_{q, deg(q)=k+p-1} \|x_j - q(A)v\| \\
&\leq \|x_j - \pi^*(A)h(A)X\xi\| \\
&= \|x_j - X\pi^*(\Lambda)\mathcal{D}^{(k)} q_j(\Lambda)\xi\| \\
&= \|x_j - X\mathcal{D}^{(k)}X^{-1}X\pi^*(\Lambda)q_j(\Lambda)\tilde{\xi}\| \\
&= \|x_j - X\mathcal{D}^{(k)}X^{-1}\pi^*(A)\phi\tilde{x}_j\| \\
&= \|x_j - X\mathcal{D}^{(k)}X^{-1}Xe_j + X\mathcal{D}^{(k)}X^{-1}Xe_j - X\mathcal{D}^{(k)}X^{-1}\pi^*(A)\phi\tilde{x}_j\| \\
&\leq \|x_j - \mathcal{D}^{(k)}_{jj} x_j\| + \|X\mathcal{D}^{(k)}X^{-1}\| \|x_j - \pi^*(A)\phi\tilde{x}_j\|. \qquad (3)
\end{aligned}$$

The result follows by choosing $\pi^* = \frac{1}{\phi}\pi_d$ where π_d is the polynomial that minimizes the distance of x_j from $\mathcal{K}_p(\tilde{x}_j)$, and assuming $\|x_j\| = 1$. □

The term $\|X\mathcal{D}^{(k)}X^{-1}\|$ is bounded as follows [28]:

$$\|X\mathcal{D}^{(k)}X^{-1}\| < k_2(X) \max_{j \neq l+1, \ldots, k'} \prod_{i=l+1}^{k'} \frac{\lambda_j - \lambda_i}{\lambda_j - \mu_i} = k_2(X)F_k,$$

where $k_2(X) = \|X\|\|X^{-1}\|$ is the condition number of the eigenvectors. If k is large enough, then the approximations μ_i converge to λ_i for $i = 1, \ldots, k'$. Thus, $F_k \to 1$, and $|1 - \mathcal{D}^{(k)}_{jj}| \to 0$. Even when these are not accurately converged, provided that $\mathcal{O}(\mathrm{dist}) < \mathcal{O}(|1 - \mathcal{D}^{(k)}_{jj}|)$, the distance behaves similarly to the distance from a deflated Krylov subspace. It should be noted that the above bound is rather pessimistic, since $\mathcal{D}^{(k)}$ converges to a part of the identity matrix and thus $X\mathcal{D}^{(k)}X^{-1}$ converges to a spectral projector.

4.1 Deflation in IRA(k, m)

Theorem 4.1, can be applied to the $k+p$ vectors at the end of an IRA(k, m) step. As previously, l eigenpairs are needed, k pairs are retained after each restart, and $p = m - k$ additional vectors are built. Theorem 4.1 applies with the same l, k, p and $k' = k$:

$$\mathrm{dist}(x_j, \mathcal{K}_{k+p}(v^{(s)})) \leq |1 - \mathcal{D}^{(k,s)}_{jj}| + \|X\mathcal{D}^{(k,s)}X^{-1}\| \ \mathrm{dist}(x_j, \mathcal{K}_p(\tilde{x}_j)).$$

Note that the space $\mathcal{K}_k(v^{(s)})$ contains exactly the wanted k Ritz vectors at the end of the previous $s-1$ step. From the comments in section 2, the Krylov space $\mathcal{K}_{k+p}(v^{(s)})$ is built implicitly by only p steps. Therefore, Theorem 4.1 relates the p steps of the deflated method, to p, rather than $k+p$ steps of the original method.

The diagonal elements of $\mathcal{D}^{(k,s)}$ depend on two parameters: k the number of the initial Krylov steps, and s the restarting step on which the theorem is applied. Since k in IRA(k,m) is bounded, the reason for convergence of $\mathcal{D}_{jj}^{(k,s)}$ is assumed by s, the step number. It has been proved for the symmetric case, and under certain assumptions for the nonsymmetric case [24], that the retained eigenpairs in IRA(k,m) converge. Thus, $F_k^{(s)} \to 1$ and $|1-\mathcal{D}_{jj}^{(k,s)}| \to 0$, as $s \to \infty$. After several restarts, the IRA(k,m) method builds a space close to the one built by an IRA(k,m) applied on a system deflated from the eigen-components $l+1,\ldots,k$. Because of Theorem 3.1, the GD(k,m) performs in a similar way.

The above results suggest that there are advantages in keeping more vectors at each restart, i.e., using a thicker restart. If only the wanted eigenpairs $(1,\ldots,l)$ are retained at restart, the method does not demonstrate the deflation behavior for any other eigenpairs. At every restarting the current approximations of eigenpairs $(l+1,\ldots,k+p)$ are annihilated, and thus they do not converge. Frequently, some eigenvalues close to the wanted ones or close to the other end of the spectrum are relatively well approximated before restarting, and if retained, they would have converged soon. Even more undesirable is the fact that these approximations will slowly reappear in the Krylov subspace, since their approximations are not accurate enough to completely annihilate the corresponding eigenvectors. Therefore, thick restarting should almost always be beneficial.

5 Dynamic thick restarting in the symmetric case

In this section we restrict the discussion to the symmetric case where explicit bounds for convergence rates are known. Two difficulties are associated with thick restarting: the choice of which eigenpairs to retain, and how many of them. It is well known that the Arnoldi method constructs vectors with strong components in the direction of the extreme eigenvectors (associated with extreme eigenvalues), and therefore close to the few wanted ones. Sleijpen et al. in [23] argue that the restarted Arnoldi method repeats the information for these extreme eigenpairs that are dispensed in previous iterations, and they propose keeping $l+1,\ldots,k$ eigenvalues closest to the wanted ones. A similar strategy is followed in the implicit restarting of the ARPACK code. We denote this special case of GD(k,m) as TR(k), implying the basis size m.

The preceding discussion suggests that thick restarting should aim at improving the convergence of the method through deflation. TR(k) attempts to increase the gap of the wanted eigenvalues from the rest of spectrum by keeping nearby eigenpairs. The same objective is followed by subspace iteration where the number of vectors determines the rate of convergence. Since IRA(k,m) can be interpreted as an efficient way to perform subspace iteration [12], similar restarting considerations hold. However, convergence depends on the gap ratios of the eigenvalues and therefore the other end of the spectrum is also of importance. A more general form of thick restarting would be TR(L,R), where L lowest (leftmost) and R highest (rightmost) eigenvectors are kept.

We need to address the issue of choosing optimal restarting parameters. In ARPACK, k is chosen dynamically, starting from a relatively small number and increasing it every time an eigenvalue converges. This attempts to maintain a “constant” gap, and it is slightly different from the strategy reported in [24], where values of k close to $m/2$ usually gave the best results.

Because of the deflation relation, the thicker the restarting, the larger the part of the spectrum that is deflated. However, the basis size m is limited, and if too many vectors are retained when restarting, the Lanczos process can not effectively build additional basis vectors. A dynamic choice of the parameters L and R should be able to capture this trade-off. For the Lanczos procedure, convergence is governed by a term involving a Chebyshev polynomial. If p Lanczos steps are taken, the error of the i_{th} eigenvalue involves the following term:

$$\frac{1}{T_p^2(1+2\gamma_i)}, \quad \text{with } \gamma_i = \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_N}.$$

γ_i is the gap ratio of the i_{th} eigenvalue, and for small gap ratios (i.e., difficult problems) the above term behaves as:

$$\frac{1}{T_p^2(1+2\gamma_i)} \approx 2e^{-2p\sqrt{\gamma_i}}. \tag{4}$$

The L and R thick restarting parameters should maximize the deflated gap ratio $\gamma_i = (\lambda_i - \lambda_{L+1})/(\lambda_{L+1} - \lambda_{N-R})$ and also maximize the number of new Lanczos steps $p = m - L - R$. The trade-off is captured by minimizing the error approximation equation (4). Since the actual eigenvalues are not known, the m approximate Ritz values (μ_i) before restarting should be used to estimate the spectrum. Thus, assuming the l lowest eigenpairs are sought, L and R are obtained dynamically by maximizing the following expression:

$$\max_{L=l,\ldots,m,\ R=0,\ldots,m-l,\ L+R<m} (m-L-R)\sqrt{\frac{\lambda_i - \lambda_{L+1}}{\lambda_{L+1} - \lambda_{m-R}}}.$$

We implement a combination of the dynamic restarting and the TR(L) schemes. Similarly to subspace iteration and ARPACK, we keep at least $L' > l$ vectors from the side of the required eigenpairs to guarantee an increased separation gap. In the experiments in the next section the value $L' = 10$ is chosen. The dynamic scheme is adopted for the rest of the vectors, maximizing the above expression for $L = L', \ldots, m$. In this way, we capture the benefits from both strategies. It has been observed that if some unwanted eigenvector has converged it is usually beneficial to include it in restarting, since this information may be slowly repeated. We do not consider this option and let the dynamic choice of L and R take care of such cases.

For the nonsymmetric GD(k,m) a similar expression may be maximized, where the Ritz values are ordered according to the required objective, i.e., largest modulus, largest real part, etc. Often, this ordering corresponds to the outermost eigenvalues of the spectrum that the Arnoldi method approximates first, and thus similar deflation arguments can be made. However, this may not always be true, and the choice is more ad-hoc because of lack of general expressions for convergence rates. The dynamic strategy can also be used in case of preconditioning, although its effects are expected to be less pronounced for two reasons. First, the spectrum of the varying operator is transformed by the preconditioners, and second the preconditioning equation usually targets one specific eigenvector for correction, offering little improvement to the rest of the eigenvectors. Often, however, the use of less efficient preconditioners does not affect the eigenvalue order significantly, and thick restarting can perform as well in this case. Finally, dynamic thick restarting can be used in both GD(k,m) and in the IRA(k,m) of the ARPACK package.

6 Numerical experiments

In the first part of this section we give a small artificial example which demonstrates the increasing effect of deflation in thick restart TR(k). In the second part, we present results from a large number of tests on the symmetric matrices of the Harwell-Boeing collection [9]. The $\mathrm{GD}(k,m)$ code is based on a program published in [25] and the extensions proposed in [26]. It implements a variable block generalized Davidson method, using the reverse communication protocol for matrix vector multiplication and preconditioning operations. Robust shifting and the Olsen strategy, which is equivalent to the Jacobi-Davidson approach in exact arithmetic [23], are adopted in preconditioning. In the third and fourth parts, the dynamic strategy is used to provide the shifts to the $\mathrm{IRA}(k,m)$ of the ARPACK implementation. Results from standard nonsymmetric cases are reported in the third part. In the last part, comparisons with the original ARPACK code, and with the ARPACK code using Leja shifts [2] in the symmetric case facilitate a discussion on the effects of the basis size.

6.1 Deflation works

The $\mathrm{GD}(k,m)$ is applied on an artificially generated diagonal matrix of order 100, and elements:

$$A_{jj} = \begin{cases} j/55, & \text{for } j = 1,\ldots,8 \\ 19/55 + j/55, & \text{for } j = 9,\ldots,16 \\ j-16, & \text{for } j = 17,\ldots,100 \end{cases} . \tag{5}$$

The lowest eigenvalues of this matrix are grouped in two clusters of 8 equidistant eigenvalues each. The separation between the two groups is equal to the separation of the second group from eigenvalue 17. Figure 1 depicts the lowest part of this spectrum. We look for the lowest eigenvalue and allow for 20 basis vectors in all versions of $\mathrm{GD}(k,m)$. The history of the logarithm of the eigenvalue error is plotted in Figure 2 for various restarting thicknesses of TR(k).

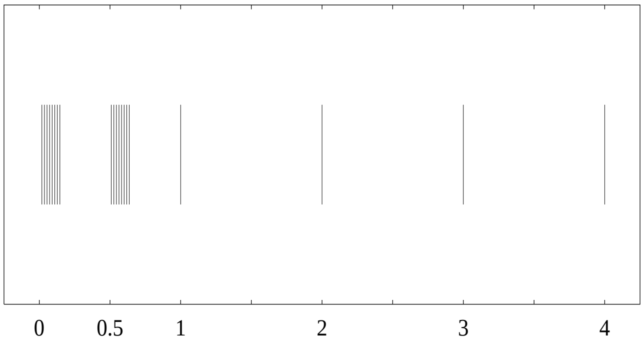


Figure 1: The lowest 20 eigenvalues of the 100×100 matrix. The first two clusters contain 8 equidistant eigenvalues each. The rest 80 eigenvalues are the integers from 5 up to 84.

As expected, the poor separation of the lowest eigenvalue results in a very slow original GD(20) (or TR(1)) method. A very good approximation of the second eigenvalue is available quite early, and thus when retained (TR(2)), the convergence rate improves by 30%. Similarly with TR(4) and TR(8). The superlinear convergence is more evident in TR(8). In early iterations, higher eigenvalues are not well approximated and TR(8) behaves similarly to TR(1) and TR(2). Later, as better approximations for eigenvalues 2–4 appear, TR(8) is similar to TR(4), and as higher eigenvalues settle down, TR(8) exhibits a concave convergence curve.

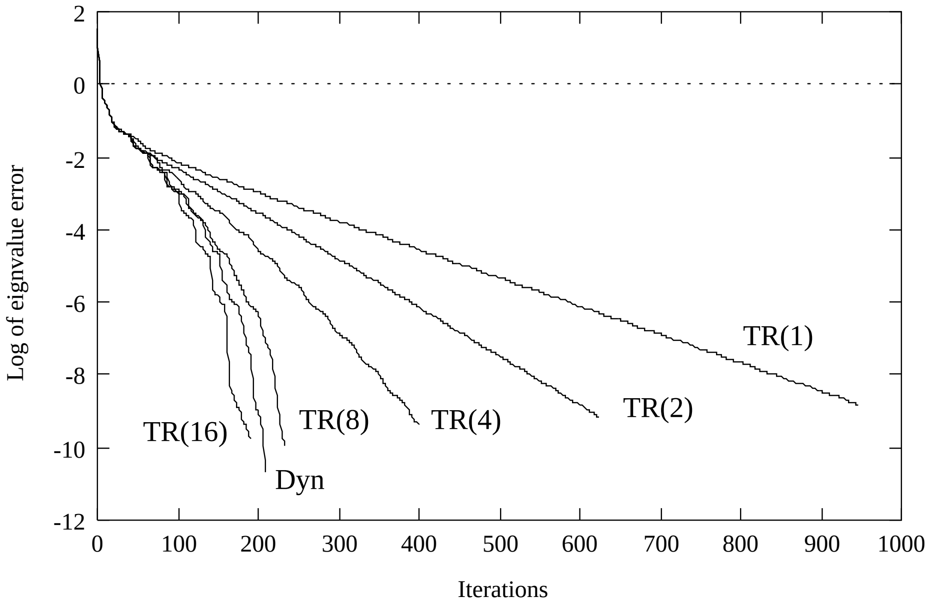


Figure 2: Effects of thick restarting to the convergence of the Generalized Davidson. No preconditioning is used, and the lowest eigenvalue is sought. TR(k) denotes GD(k,20).

Methods TR(k), with $8 < k < 16$ are similar to TR(8) since there is no significant improvement to the deflated gap ratio. In theory, TR(16) should be different, because of the large separation between eigenvalues 16 and 17. In practice however, TR(16) does not perform significantly better than TR(8). The reason is that the Krylov subspace is of dimension 20, and it is difficult for the 16_{th} Ritz eigenvalue to converge. The dynamic thick restarting, shown as Dyn in the figure, takes advantage of both ends of the spectrum and performs better than TR(8) and close to TR(16), requiring no prior knowledge about the spectrum.

6.2 Harwell-Boeing Tests

To confirm the theoretical benefits of thick and dynamic thick restarting, a wide variety of tests has been performed on the symmetric matrices from the Harwell-Boeing collection. This includes a set of 67 matrices with orders ranging from 48 to 15,439. Some of matrices have been derived from eigenvalue problems, but for almost all of them, the lowest end of spectrum is very poorly conditioned, making them particularly hard test problems. The higher end of the spectrum usually consists of well separated, very large eigenvalues, providing a good test for easy or intermediate problems.

We have compared three different versions of GD(k, m) for both the lower and the higher part of the spectrum. Five eigenvalues are sought and the basis size m for all GD methods is

20. An eigenpair is considered converged when the norm of its residual is less than $10^{-12}\|A\|_F$, where $\|A\|_F$ is the Frobenius norm of its matrix. For the highest eigenvalues only the non-preconditioned versions of $\mathrm{GD}(k,m)$ are considered, while for the lowest ones we consider diagonal, and approximate inverse preconditioning. The former is computed at every step as $(\mathrm{diag}(A)-\mu)^{-1}$, and the latter is only computed once as the approximate inverse of A [4]. Since most of the matrices are positive definite, this is a relatively powerful preconditioner.

In Table 1, the results from the lower part of the spectrum are reported. A maximum number of 5000 matrix vector multiplications is allowed. The table does not include any of the diagonal matrices. As it is easily seen, TR(11) outperforms the original Davidson method (TR(5)), except for BCSSTK22. It is usually several times faster, and offers better robustness, converging for 6 additional matrices. Further, dynamic thick restarting, improves both the robustness, and the speed in almost all cases. Sometimes, the reduction in the matrix vector multiplication number can be up to 50-70% over TR(11). With diagonal preconditioning TR(11) still outperforms TR(5) in both convergence and robustness. Dynamic thick restarting improves convergence even further, although the improvements are not as impressive as in the non-preconditioned case. On the average, the approximate inverse preconditioner is better than the diagonal one, but with several exceptions since it depends on the characteristics of the matrix. Dynamic thick restarting still performs much better than the original approach, and it is relatively faster and more robust than TR(12). However, as mentioned in the previous section, in those cases where approximate inverse works well the differences between thick and dynamic thick restarting diminish, because of the higher quality preconditioner.

Similar behavior of the methods is shown in Table 2, where the five largest eigenpairs are required. Dynamic thick restarting improves on the performance of TR(10) which in turn improves on the performance of TR(5). However, the few steps required for the problems in this table do not yield the same impressive improvements as in Table 1.

6.3 The effect of the basis size

The dynamic thick restarting strategy, developed for the $\mathrm{GD}(k,m)$, can also be used to provide the shifts to the ARPACK code through the supplied reverse communication protocol. Results from this implementation when seeking one lowest eigenpair of the Harwell-Boeing collection appear in Table 3. Two tests are performed, one with basis size of 25, and one with basis size of 10. The dynamic restarting significantly improves the speed and robustness of the native restarting scheme of ARPACK, which for one eigenvalue is the equivalent with thick restart of half the basis size. What is more interesting is that dynamic restarting seems much less sensitive to reduction of the basis size. Similar insensitivity to the basis size has recently been demonstrated through the use of Leja points as shifts in $\mathrm{IRA}(k,m)$ [2]. We have implemented the Leja shifts restarting strategy as outlined in [2], and the results appear in Table 3. For the small basis size, dynamic thick restarting and Leja shifts are comparable. However, as the basis size increases, the dynamic strategy is more efficient and even more robust. Although Leja shifts may be better for extremely small spaces (less than 5 vectors), they are harder to implement and they are more expensive to compute.

Experience with the dynamic thick restarting has shown that most of the vectors are retained at every restart, and only 3 or 4 are annihilated. The range of the annihilated ones varies from step to step. Figure 3 shows the range of eigenvalues which the filtering polynomial covers, as well as the shifts of this polynomial, at every restart for a typical case. We have observed that it

Table 1: Comparison of thick (TR(L)) and dynamic thick restarting (Dyn) with original Davidson (TR(5)) on symmetric Harwell-Boeing matrices, with diagonal and approximate inverse preconditioners. The number of matrix vector multiplications is reported, with a maximum of 5000. Five smallest eigenvalues are sought. The GD codes use basis size of 20.

	No preconditioning			Diagonal preconditioning			Approximate Inverse		
Matrix	TR(5)	TR(11)	Dyn	TR(5)	TR(10)	Dyn	TR(5)	TR(12)	Dyn
BCSSTK01	-	1675	360	288	132	124	264	96	108
BCSSTK02	-	209	204	-	194	190	188	89	92
NOS4	321	178	171	405	261	244	127	90	91
BCSSTK03	-	-	-	-	3697	1225	-	4699	1685
BCSSTK04	-	-	1905	-	189	188	-	208	221
BCSSTK22	4054	-	1626	-	931	721	-	320	300
LUND A	-	2017	727	858	271	250	3623	394	349
LUND B	-	-	1347	774	396	349	909	381	338
BCSSTK05	1174	975	612	1322	465	409	358	247	251
BCSSTK07	-	-	-	-	-	1401	-	-	3158
BCSSTM07	-	-	3171	1018	406	363	-	2390	1195
NOS5	-	2016	921	2659	1401	819	837	387	354
662 BUS	-	-	-	3220	1482	902	699	307	291
NOS6	-	-	-	-	-	1434	-	-	-
685 BUS	-	-	1793	2473	987	763	486	272	267
NOS7	-	-	-	200	216	194	128	109	94
GR 30 30	259	228	229	248	224	221	204	146	143
NOS3	2179	620	458	2096	878	664	524	253	258
BCSSTK09	-	1206	721	2283+	1508	964	3291	363	352
BCSSTK10	-	-	-	-	-	2808	-	2093	1076
BCSSTM10	498	226	207	448	258	250	3266	3189	2636
BCSSTK27	-	-	-	-	-	3307	-	-	3017
BCSSTM27	-	4455	1689	-	4304	1768	-	636	509
BCSSTK14	-	-	-	-	-	2136	-	-	3723
BCSSTM13	-	-	-	381	285	269	291	183	177
BCSSTK21	-	-	-	-	2568+	1141	1776	877	601
BCSSTK16	3962	1333	676	2410	905	663	752	331	317
BCSSTK18	-	-	-	-	-	3098	-	-	-
BCSSTM25	-	-	-	62	64	55	40	38	37

+ denotes that one eigenpair has been skipped

Table 2: Comparison of thick (TR(10)) and dynamic thick restarting (Dyn) with original Davidson (TR(5)) on Harwell-Boeing matrices. The number of matrix vector multiplications is reported. Five largest eigenvalues are sought. The GD codes use basis size of 20.

	No preconditioning		
Matrix	TR(5)	TR(10)	Dyn
BCSSTK01	57	42	38
BCSSTK02	62	49	52
NOS4	176	107	114
BCSSTK03	51	44	43
BCSSTK04	103	84	78
BCSSTK22	106	71	65
LUND A	195	124	120
LUND B	92	66	68
BCSSTK05	81	67	66
NOS1	257	147	133
PLAT362	165	111	114
BCSSTK06	332	114	109
BCSSTK07	332	114	109
BCSSTM07	240	172	155
NOS5	210	117	111
662 BUS	65	54	55
NOS6	123	91	87
685 BUS	31	30	30
NOS7	88	65	68
BCSSTK19	113	100	92
GR 30 30	502	451	396

	No preconditioning		
Matrix	TR(5)	TR(10)	Dyn
NOS2	2236	906	520
NOS3	194	156	150
BCSSTK08	36	35	33
BCSSTK09	316	236	206
BCSSTK10	146	94	90
BCSSTM10	443	151	137
1138 BUS	84	73	75
BCSSTK27	129	89	81
BCSSTM27	130	96	87
BCSSTK11	441	220	200
BCSSTM12	164	115	129
BCSSTK14	195	73	75
PLAT1919	102	94	100
ZENIOS	53	48	48
BCSSTK24	112	118	121
BCSSTK21	1144	418	335
BCSSTK15	-	1374	328
BCSSTK16	99	83	83
BCSSTK17	82	67	62
BCSSTK18	166	86	86
BCSSTK25	45	59	44

Table 3: Implementation of the Leja shifts and dynamic thick restarting for the ARPACK code. Native is the restarting scheme used internally by ARPACK, Leja(k) refers to implicit restarting with k Leja shifts, and Dyn is the dynamic thick restarting. The number of matrix vector multiplications is reported for two tests with basis sizes 10 and 25, on Harwell-Boeing matrices. One lowest eigenvalue is sought.

	ARPACK					
	Basis Size of 10			Basis Size of 25		
Matrix	Native	Leja(3)	Dyn	Native	Leja(5)	Dyn
BCSSTK01	-	3805	3922	1637	1309	341
BCSSTK02	530	235	198	129	134	124
NOS4	220	136	166	116	114	120
BCSSTM03	1165	1696	298	265	1014	90
BCSSTK04	-	-	-	-	-	2013
BCSSTK22	-	1132	1222	1520	1124	999
BCSSTM22	240	166	149	103	104	89
LUND A	-	2461	1644	2079	1774	759
LUND B	-	1990	2777	3002	1404	1150
BCSSTK05	2810	727	874	766	609	588
BCSSTM06	4675	1462	494	792	529	243
NOS5	-	1123	1546	1494	864	880
BCSSTM20	-	-	-	-	-	896
494 BUS	-	-	-	-	-	3634
662 BUS	-	1642	1547	2443	1429	1108
685 BUS	-	2482	2515	1962	1819	700
NOS3	1210	388	492	402	334	348
BCSSTK09	1140	367	419	337	309	304
BCSSTM10	420	214	274	181	164	174
BCSSTM27	-	2698	1931	2781	1509	1461
BCSSTM11	65	196	41	25	25	25
BCSSTM13	-	4285	3471	-	4459	2995
ZENIOS	60	58	56	90	84	80
BCSSTK16	-	946	1063	1000	774	712
BCSSTK25	-	-	-	-	-	1399

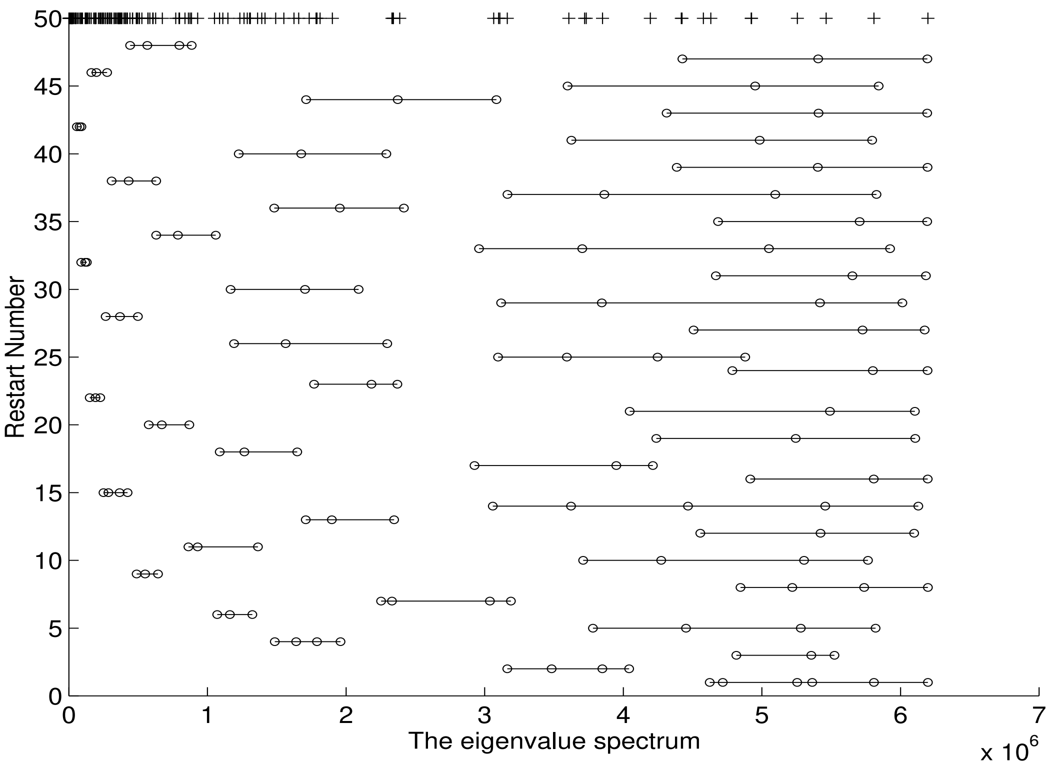


Figure 3: The range annihilated by the filtering polynomial of dynamic thick restarting at every restart. Each interval includes the m-L-R Ritz values, depicted as circles, which are picked for annihilation by the dynamic scheme. The example matrix is BCSSTK05 from Harwell-Boeing and basis size of 20 is used. The crosses on the top of the graph represent the location of the eigenvalues on the real axis.

is important to have both a small degree polynomial at every restart (i.e., only few eigenvalues annihilated) and to also vary the range from where these shifts are chosen. Therefore, TR(16) does not perform as well as dynamic restarting, even though, on average, it retains the same number of vectors. Also, if we force the dynamic restarting to annihilate more than 5 or 6 shifts at every restart, the scheme does not perform as well either. The efficiency of the dynamic thick restarting may be attributed to the fact that the filtering polynomial is of low degree and seems to select the best region to dampen, without growing fast outside these regions. The efficient use of the Leja shifts in the ARPACK also exhibits analogous requirements.

Finally, we should point out that the above results compare the number of matrix vector multiplications of the methods. This is an acceptable performance metric if the matrix-vector operation is expensive. Since on average, thick restarting uses more vectors in the basis than the original Davidson, its Davidson step is also more expensive. Although improvements like the ones in Table 1 justify any increase in the expense of the Davidson step, for easier cases a less aggressive choice of restarting might be more effective.

6.4 Thick restarting in the nonsymmetric case

As in the symmetric case, we can use likewise the dynamic thick restarting scheme to provide the shifts to the nonsymmetric ARPACK code. Results from this implementation applied on the nonsymmetric matrices of the test matrix collection of eigenvalue problems of Bai et al. [1] appear in Table 4. All the matrices stem from standard eigenvalue problems, except ODEP400A which is included because it is close to symmetric. Since for almost all examples the rightmost eigenpairs are of interest, we look for five eigenpairs with largest real parts. The convergence threshold for ARPACK is set to 10^{-12}, and a maximum of 5000 matrix vector multiplications is allowed.

The shifts for thick restarting are chosen similarly to the symmetric case. First, we order the Ritz values according to their real parts. The dynamic scheme works on these real parts, yielding the numbers L and R on the real axis. We then supply the corresponding Ritz values as shifts to ARPACK, requiring that conjugate Ritz values are either annihilated together or kept together.

The results show that the thick restarted versions improve efficiency and robustness of the native scheme of ARPACK, and that thicker restarting schemes achieve better efficiencies. This is expected by analogy with the subspace iteration method. The dynamic thick restarting is not uniformly better than the rest as in the symmetric case. In fact, it seems comparable to TR(20) which on the average keeps the same number of vectors as the dynamic one. As mentioned in section 5, the extreme eigenpairs chosen by the dynamic scheme are based on the ordering of the real parts of the Ritz values and may not always represent the extreme eigenpairs approximated well by the Arnoldi method. In spite of this, dynamic thick restarting is still the most robust of the methods used, and shows that the efficiency of the one-sided thick restarting can be improved.

7 Conclusions

Restarting is a necessary technique for solving large eigenvalue problems, which may cause significant convergence deterioration. In this paper we consider a class of restarting techniques

Table 4: Implementation of thick and dynamic thick restarting for the nonsymmetric ARPACK code. Native is the restarting scheme used internally by ARPACK, TR(L) is one-sided thick restarting with L vectors, and Dyn is the dynamic thick restarting. The number of matrix vector multiplications is reported for the test-matrix collection for eigenvalue problems. Five eigenvalue with largest real parts are sought.

	ARPACK			
Matrix	Native	TR(12)	TR(20)	Dyn
BWM200	558	207	185	180
BWM2000	-	-	-	3999
CDDE5	357	272	252	237
DW2048	681	675	815	495
DW8192	-	-	-	4942
DWA512	118	116	116	113
DWB512	350	298	315	267
GRCAR200	2606	698	524	572
LOP163	383	279	214	242
ODEP400A	1683	837	1005	704
OLM100	548	357	255	316
OLM1000	-	-	-	3602
OLM500	4303	2514	1867	1622
PDE225	343	281	234	254
PDE2961	192	140	124	130

	ARPACK			
Matrix	Native	TR(12)	TR(20)	Dyn
QH768	-	2935	751	881
RDB1250	610	454	145	139
RDB1250L	524	513	436	449
RDB2048	887	181	185	170
RDB2048L	755	615	588	598
RDB3200L	842	738	736	729
RDB450	376	259	90	85
RDB450L	343	295	280	309
RDB800L	429	421	354	391
RW136	170	128	109	108
RW496	247	179	164	168
RW5151	743	514	406	473
TOLS90	-	-	330	1295
TUB100	318	181	154	165
TUB1000	-	4042	3730	1696

which, at every restart, retain more Ritz vectors than needed, and we denote it as 'thick restarting'. The GD(k,m) and IRA(k,m) are proved to be equivalent in the absence of preconditioning and a relation is given between thick restarted Davidson, and a Davidson method applied on a deflated system. These theoretical results imply that retaining more outermost Ritz pairs can enhance convergence.

For the symmetric case, the results can be interpreted as an effort to increase the gap ratio for the required eigenvalues. Since the number of basis vectors is limited, the actual objective is to maximize the error reduction between restarts. This gives rise to a dynamic thick restarting technique which applies to IRA(k,m) and to the preconditioned GD(k,m). The extensive numerical experiments demonstrate the efficiency and robustness of the dynamic thick restarting, and show that the robustness carries over to the nonsymmetric case. In addition, this scheme seems to be much less sensitive to smaller Krylov subspace dimensions, and can be extremely beneficial in very large eigenvalue problems.

Acknowledgements

We are greatful to the referees and to H. A. Van der Vorst whose insightful comments improved the presentation significantly. We also acknowledge R. B. Lehoucq for long, revealing discussions on implicit restarting, as well as E. Chow for kindly providing us with his approximate inverse preconditioning code.

References

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

[2] D. Calvetti, L. Reichel, and D. Sorensen. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Electronic Trans. Numer. Anal.*, 2:1–21, 1994.

[3] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. Technical Report 95-181, University of Minnesota, Supercomputing Institute, 1995.

[4] Edmond Chow and Yousef Saad. Approximate inverse preconditioners via sparse-sparse iteration. *SIAM J. Sci. Comput.*, To appear.

[5] P. Concus, G. H. Golub, and D. P. O'Leary. *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations.* Academic Press, New York, NY, 1976.

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

[7] J. Cullum and R. A. Willoughby. *Lanczos algorithms for large symmetric eigenvalue computations*, volume 2: Programs of *Progress in Scientific Computing; v. 4*. Birkhauser, Boston, 1985.

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

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

[10] 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. Technical Report 941, Department of Mathematics, University of Utrecht, 1996. To appear in SISC.

[11] N. Kosugi. Modification of the Liu-Davidson method for obtaining one or simultaneously several eigensolutions of a large real-symmetric matrix. *J. Comput. Phys.*, 55:426–436, 1984.

[12] R. B. Lehoucq. *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration.* PhD thesis, Department of Computational and Applied Mathematics, Rice University, 1995. TR95-13.

[13] R. B. Lehoucq, D. C. Sorensen, and P. Vu. An implementation of the implicitly restarted Arnoldi iteration that computes some of the eigenvalues and eigenvectors of a large sparse matrix. Technical report, University of Tennessee, Netlib, 1995.

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

[15] R. B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Math. Comput.*, 1996.

[16] 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.

[17] 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.

[18] Yousef Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Math. Comp.*, 42:567–588, 1984.

[19] Yousef Saad. *Numerical methods for Large eigenvalue problems.* Manchester University Press, 1993.

[20] Yousef Saad. Analysis of augmented Krylov subspace methods. Technical Report 95-176, University of Minnesota, Supercomputing Institute, Minneapolis, MN, 1995.

[21] Miloud Sadkane. Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems. *Numer. Math.*, 64(2):195–211, 1993.

[22] D. S. Scott. The advantages of inverted operators in Rayleigh-Ritz approximations. *SIAM J. Sci. Comput.*, 3:102, 1982.

[23] 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), 1996.

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

[25] Andreas Stathopoulos and Charlotte 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.

[26] Andreas Stathopoulos, Yousef Saad, and Charlotte Fischer. Robust preconditioning of large, sparse, symmetric eigenvalue problems. *Journal of Computational and Applied Mathematics*, 64:197–215, 1995.

[27] A. Van der Sluis and H. A. Van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48:543–560, 1986.

[28] H. A. Van der Vorst and C. Vuik. The superlinear convergence behavior of GMRES. *Journal of computational and applied mathematics*, 48:327–341, 1993.

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