ABSTRACT

Restarting the Lanczos Algorithm for Large Eigenvalue Problems
and Linear Equations

Dywayne A. Nicely, Ph.D.

Advisor: Ronald B. Morgan, Ph.D.

We are interested in computing eigenvalues and eigenvectors of large matrices and in solving large systems of linear equations. Restarted versions of both the symmetric and nonsymmetric Lanczos algorithms are given.

For the symmetric case, we give a method called Lan-DR that simultaneously solves linear equations and computes eigenvalues and eigenvectors. The use of approximate eigenvectors deflates eigenvalues. Maintaining the orthogonality of the Lanczos vectors is a concern. We suggest an approach that is a combination of Parlett and Scott's idea of selective orthogonalization and Simon's partial orthogonalization. For linear systems with multiple right-sides, eigenvectors computed during the solution of the first right-hand side can be used to give much faster convergence of the second and subsequent right-hand sides.

A restarted version of the nonsymmetric Lanczos algorithm is developed. Both the right and left eigenvectors are computed while systems of linear equations are solved. We also investigate a restarted two-sided Arnoldi. We compare expense and stability of this approach with restarted nonsymmetric Lanczos.

Restarting the Lanczos Algorithm for Large Eigenvalue Problems
and Linear Equations

by

Dywayne A. Nicely, B.S., M.A.

A Dissertation

Approved by the Department of Mathematics

Lance L. Littlejohn, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Doctor of Philosophy

TABLE OF CONTENTS

ACKNOWLEDGMENTS

First and foremost, I would like to thank my parents and two brothers for always believing in me. I am ever grateful of their love and support as I continued my education. I would like to thank Baylor University for allowing me to pursue my doctorate. For preparing me to do research, I would like to thank the Baylor mathematics department faculty. In particular, I would like to thank Dr. Markus Hunziker for his LaTex help. Also I would like to thank Dr. Johnny Henderson for the many conversations we had. Lastly, I would like to thank Dr. Ronald Morgan for his guidance and being a true mentor in the process of this dissertation.

CHAPTER ONE

Introduction

In this dissertation, we are interested in solving the eigenvalue problem

$$Az = \lambda z,$$

and the linear equations problem

$$Ax = b.$$

The cases where A is a large symmetric (and Hermitian) and a large nonsymmetric (and non-Hermitian) square matrix are discussed. We deflate eigenvalues using the thick restarting technique given by Wu and Simon in [25, 56, 64] and as mentioned both new methods are related to implicitly restarted Arnoldi [56] and GMRES-DR [25]. Deflating eigenvalues can improve the convergence of restarted methods. The idea of deflation is as follows. At the restart we augment the Krylov subspace with approximate eigenvectors. This allows us to compute many eigenvalues at the same time. For linear equations, the approximated eigenvectors essentially remove or deflate the corresponding eigenvalues. This can dramatically improve the convergence and can mostly make up for the effect of the restarting.

The new methods are called Lanczos with deflated restarting (Lan-DR) and Nonsymmetric Lanczos with deflated restarting (NLan-DR), respectively. When solving systems of linear equations Lan-DR is closely related to the Conjugate Gradient method (CG) while NLan-DR it is closely related to the Biconjugate Gradient method (BiCG). Convergence theories on how deflation aides Lan-DR and NLan-DR are given. Also due to the known instabilities of the Lanczos algorithm, we give an alternate method to NLan-DR. This method is Two-Sided Arnoldi with deflated restarting (TSArn-DR). TSArn-DR is also equipped to solve systems of linear equations.

Also, in the case of a linear system with multiple right-hand sides the deflation of small eigenvalues aides in the quicker convergence of the second and subsequent right-hand sides. An application involving multiple right-hand sides is Quantum Chromodynamics problems or QCD problems. Some examples are given for the hermitian case. In the nonsymmetric case, we also give examples of how deflation aides in the convergence of linear systems with multiple right-hand sides.

In chapter 2, we introduce Krylov subspace methods in particular symmetric and nonsymmetric Lanczos. This chapter also includes some examples showing how polynomials can be used to analyze convergence and also includes convergence theory that uses polynomials to analyze convergence. The importance of deflation in Krylov methods is discussed. Lan-DR is presented in Chapter 3 and numerical examples are given for solving the eigenvalue problems and finding solutions to systems with multiple right-hand sides (this includes QCD problems). Chapter 4 presents NLan-DR, and NLan-DR's linear equation solvers, biconjugate gradient with deflated restarting (BiCG-DR) and deflated biconjugate gradient stabilized (D-BiCGStab). Numerical examples are given including comparisons to ARPACK and BiCG-DR versus GMRES-DR (first right-hand side) and non-deflated BiCGStab and D-BiCGStab (for the second and subsequent right-hand sides) for linear equations. In chapter 5, TSArn-DR is discussed and examples comparing NLan-DR and TSArn-DR are given.

CHAPTER TWO

Preliminaries

2.1 The Rayleigh-Ritz Procedure for eigenvalues

Standard eigenvalue methods such as the QR iteration are too expensive for large problems. So iterative methods are need for these large problems, and of those iterative methods, Krylov subspace methods are the most popular.

The Rayleigh-Ritz procedure,[38, 46] extracts approximate eigenvectors from a subspace of $\mathbb{R}^n$ by reducing to a smaller eigenvalue problem. The Krylov methods described later use this procedure.

The Rayleigh-Ritz Procedure

1. Let S be a $j-$dimensional subspace of $\mathbb{R}^n$.
2. Compute Q, an $n \times j$ orthonormal matrix whose columns span S.
3. Compute the $j \times j$ matrix $H = Q^T AQ$.
4. Find eigenvalues θ_i of H, and if desired, find eigenvectors g_i of unit length. The θ_i are approximate eigenvalues of A called Ritz values. The Ritz vectors, $y_i = Qg_i$, are approximate eigenvectors of A. The residual norms are $\|r_i\| = \|Ay_i - \theta_i y_i\|$.

2.2 Krylov Subspaces

The Krylov subspace of dimension m with starting vector v is:

$$K_m = K_m(A, v) = Span\{v, Av, A^2v, ..., A^{m-1}v\}.$$

A Krylov subspace has the following property: K_m is the subspace of all vectors in $\mathbb{R}^n$ which can be written as $x = p(A)v$ where p is a polynomial degree $m - 1$ or less.

2.3 Krylov Methods

In this section we discuss methods that utilize Krylov subspaces. These methods are very useful for finding approximations to eigenvalues and can be used to solve systems of linear equations. We first focus on eigenvalues.

2.3.1 Arnoldi Method

The Arnoldi method [2, 44, 46] is used for a large matrix A. The method produces a small matrix H_m which can give accurate approximations to some of the eigenvalues of A. We offer the following algorithm which is actually the Modified Gram-Schmidt version of the Arnoldi algorithm. This method is more reliable than the standard version of the Arnoldi method in the face of roundoff problems.

The Arnoldi Algorithm

```
1. Choose an initial vector $v_1$ of norm 1.
2. For $j = 1, 2, ...m$. Do:
3. Compute $w_j := Av_j$
4. For $i = 1, 2, ...j$. Do:
5. $h_{ij} := (w_j, v_i)$
6. $w_j := w_j - h_{ij}v_i$
7. EndDo
8. $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ Stop
9. $v_{j+1} = w_{j+1}/h_{j+1,j}$
10. EndDo
```

The resulting H_m matrix is upper-Hessenberg. An upper-Hessenberg matrix has zero entries if $i \geq j+1$, where i and j are the row and column of the matrix. So it is an upper triangular matrix with the addition of the sub-diagonal. Also from the Arnoldi algorithm we obtain the following relations:

$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^T \\ AV_m &= V_{m+1} \bar{H}_m \\ V_m^T A V_m &= H_m \end{aligned}$$

where V_m is an $n \times m$ matrix whose columns are the vectors $v_1, ..., v_m$, $\bar{H}_m$ is the $(m+1) \times m$ upper-Hessenberg matrix whose entries, h_{ij}, are obtained from the algorithm, and H_m is obtained from removing the $m+1st$ row of $\bar{H}_m$. These relations are important, for this work, because the symmetric and nonsymmetric Lanczos algorithms have similar relations with their matrix T_m.

As stated earlier, the above algorithm is the modified Gram-Schmidt version of the Arnoldi algorithm. And while this method is often more reliable for maintaining orthogonality between the vectors, sometimes cancellations in the orthogonalization steps cause roundoff error and loss of orthogonality. The idea to remedy this is to do double orthogonalization, or a reorthogonalization, but this is a significant increase of expense for an already expensive method. It is known that reorthogonalizing more than once is redundant. Also, sometimes a partial reorthogonalization is performed where a vector is only reorthogonalized if its norm drops by over 90% during orthogonalization [5].

2.3.2 Symmetric Lanczos

The symmetric Lanczos algorithm is a special case or simplification of the Arnoldi algorithm that is used when the matrix A is symmetric.

The Symmetric Lanczos Algorithm

```
1. Choose an initial vector $v_1$ of norm 1. Set $\beta_1 \equiv 0$ and $v_0 \equiv 0$.
2. For $j = 1, 2, ...m$. Do:
3. $w_j := Av_j - \beta_j v_{j-1}$
4. $\alpha_j := (w_j, v_j)$
5. $w_j = w_j - \alpha_j v_j$
6. $\beta_{j+1} = \|w_j\|_2$ If $\beta_{j+1} = 0$ Stop
7. $v_{j+1} = w_{j+1}/\beta_{j+1}$
8. EndDo
```

Since A is symmetric, the resulting $V_m^T A V_m$ matrix, now called T_m is now tridiagonal, i.e.

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & . & . & . & & \\ & & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & & \beta_m & \alpha_m \end{pmatrix}$$

where the $\alpha_i's$ and $\beta_i's$ come from the algorithm. An orthonormal set of vectors, $\{v_1, v_2, ..., v_m\}$, results from the symmetric Lanczos algorithm. But this set of vectors has a unique property. It can be generated using a three-term recurrence. This brings us to the following proposition.

Proposition 2.1. *Given the fact that span*$\{v_1, ..., v_i\} = $ *span*$\{v_1, Av_1, ..., A^{i-1}v_1\}$ *for* $i < j$ *and* $\{v_1, v_2, ..., v_j\}$ *is a set of orthonormal vectors, if you form the next vector,* v_{j+1}, *by orthogonalizing* Av_j *against the previous two vectors,* v_j *and* v_{j-1}, *then* v_{j+1} *is automatically orthogonal to* $\{v_1, ..., v_{j-2}\}$.

Proof. Consider $w = Av_j - \alpha v_j - \beta v_{j-1}$, with α and β chosen so that w is orthogonal to v_j and v_{j-1}. We want $v_i^T w = 0$ for $i < j-1$. So,

$$\begin{aligned} v_i^T w &= v_i^T (Av_j - \alpha v_j - \beta v_{j-1}) \\ &= v_i^T Av_j - \alpha v_i^T v_j - \beta v_i^T v_{j-1} \\ &= v_i^T Av_j \end{aligned}$$

since $v_i \perp v_j, v_{j-1}$. Note,

$$span\{v_1, v_2, ..., v_j\} = span\{v_1, Av_1, ..., A^{j-1}v_i\}.$$

Therefore

$$\begin{aligned} v_i^T Av_j &= v_i^T A^T v_j (A\ symmetric) \\ &= (Av_i)^T v_j \\ &= 0 \end{aligned}$$

since $Av_i \in span\{v_1, ..., v_{i+1}\}$ and $i+1 < j$. □

As mentioned earlier, the symmetric Lanczos method has similar relations to the Arnoldi method. The difference is in the resulting matrices (H_m for Arnoldi and T_m for Lanzos):

$$\begin{aligned} AV_m &= V_m T_m + w_m e_m^T \\ AV_m &= V_{m+1} \bar{T}_m \\ V_m^T AV_m &= T_m. \end{aligned}$$

The symmetric Lanczos method can incur breakdown problems, namely where $\beta_{j+1} = 0$. However, for the symmetric Lanczos algorithm, breakdown is an advantage. Like the Arnoldi algorithm, at this breakdown step eigenpairs are exact. Then from this point one needs to restart in order to find the remaining desired eigenpairs.

2.3.3 Nonsymmetric Lanczos

The nonsymmetric Lanczos algorithm utilizes two Krylov subspaces,

$$K_m(A, v_1) = span\{v_1, Av_1, ..., A^{m-1}v_1\}$$
$$K_m(A^T, w_1) = span\{w_1, A^T w_1, ..., (A^T)^{m-1}w_1\}.$$

Biorthogonal sets of vectors, $v_i^\prime s$ and $w_i^\prime s$ respectively, are produced that span these two subspaces. By biorthogonality we mean that,

$$(v_i, w_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

which, in actuality, makes these sets of vectors biorthonormal.

The matrix A is associated with the v vectors and the w vectors with its transpose, A^T. The nonsymmetric Lanczos algorithm gives a three-term recurrence for generating the biorthonormal bases. So Orthogonalizations are saved, compared to the Arnoldi method, which reduces expense for sparse matrices and computes both right and left eigenvectors simultaneously.

The Nonsymmetric Lanczos Algorithm

```
1. Choose two vectors $v_1$ and $w_1$ such that $(v_1, w_1) = 1$.
2. Set $\beta_1 = \delta_1 \equiv 0$ and $w_0 = v_0 \equiv 0$.
3. For $j = 1, 2, ...m$. Do:
4. $\alpha_j := (Av_j, w_j)$
5. $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
6. $\hat{w}_{j+1} = A^T v_j - \alpha_j w_j - \delta_j w_{j-1}$
7. $\delta_{j+1} = |(\hat{v}_{j+1}, \hat{w}_{j+1})|^{\frac{1}{2}}$. If $\delta_{j+1} = 0$ Stop.
```

```
8. $\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})/\delta_{j+1}$

9. $w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$

10. $v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$

11. EndDo
```

The form of the resulting tridiagonal matrix, $T_m = W_m^T A V_m$, formed during the nonsymmetric Lanczos algorithm is

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \delta_2 & \alpha_2 & \beta_3 & & & \\ & . & . & & . & \\ & & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & & \delta_m & \alpha_m \end{pmatrix}$$

where the $\alpha's$, $\beta's$ and $\delta's$ come from the three-term recurrence produced by the Nonsymmetric Lanczos algorithm. The three-term recurrence is as follows,

$$v_{j+1} = \frac{Av_j - \alpha_j v_j - \beta_j v_{j-1}}{\delta_{j+1}}$$
$$w_{j+1} = \frac{A^T w_j - \alpha_j w_j - \delta_j w_{j-1}}{\beta_{j+1}}.$$

And this brings us to the following proposition.

Proposition 2.2. *Given the fact that span*$\{v_1, ..., v_i\} =$ *span*$\{v_1, Av_1, ..., A^{i-1}v_1\}$ *and span*$\{w_1, ..., w_i\} =$ *span*$\{w_1, A^T w_1, ..., (A^T)^{i-1}w_1\}$ *for* $i < j$ *,and for biorthonormal sets of vectors* $\{v_1, ..., v_j\}$ *and* $\{w_1, ..., w_j\}$*, if you form the next vectors,* v_{j+1} *and* w_{j+1}*, corresponding to the respective sets and orthogonalize* Av_j *against the previous two vectors,* w_j *and* w_{j-1}*, and do the same with* $A^T w_j$ *against* v_j *and* v_{j-1} *then* v_{j+1} *will be automatically orthogonal to* $\{w_1, ..., w_{j-2}\}$ *as will* w_{j+1} *be automatically orthogonal to* $\{v_1, ..., v_{j-2}\}$.

Proof. Let $\{v_1, ..., v_j\}$ and $\{w_1, ..., w_j\}$ be orthogonal bases. Consider $u = Av_j - \alpha v_j - \beta v_{j-1}$ with α and β chosen so that u is orthogonal to w_j and w_{j-1}. We want $w_i^T u = 0$ for $i < j - 1$.

$$\begin{aligned} w_i^T u &= w_i^T (Av_j - \alpha v_j - \beta v_{j-1}) \\ &= w_i^T Av_j - \alpha w_i^T v_j - \beta w_i^T v_{j-1} \\ &= w_i^T Av_j. \end{aligned}$$

Note, $\mathit{span}\{w_1, ..., w_j\} = \mathit{span}\{w_1, A^T w_1, (A^T)^2 w_1, ..., (A^T)^{j-1} w_1\}$. Thus,

$$\begin{aligned} w_i^T Av_j &= (A^T w_i)^T v_j \\ &= 0. \end{aligned}$$

This comes from the fact that $A^T w_i \in \mathit{span}\{w_1, ..., w_{i+1}\}$ and $\{w_1, ..., w_{i+1}\}$ are all orthogonal to v_j with $i + 1 < j$. Now consider $y = A^T w_j - \gamma w_j - \eta w_{j-1}$ with γ and η chosen such that y is orthogonal to v_j and w_{j-1}. Similarly we want $v_i^T y = 0$ for $i < j - 1$. So,

$$\begin{aligned} v_i^T y &= w_i^T (A^T w_j - \gamma w_j - \eta w_{j-1}) \\ &= v_i^T A^T w_j - \gamma v_i^T w_j - \eta v_i^T w_{j-1} \\ &= v_i^T A^T w_j. \end{aligned}$$

Note we have, $\mathit{span}\{v_1, ..., v_j\} = \mathit{span}\{v_1, Av_1, A^2 v_1, ..., A^{j-1} v_1\}$. Therefore,

$$\begin{aligned} v_i^T A^T w_j &= (Av_i)^T w_j \\ &= 0. \end{aligned}$$

This comes from the fact that $Av_i \in \mathit{span}\{v_1, ..., v_{i+1}\}$ and $\{v_1, ..., v_{i+1}\}$ are all orthogonal to w_j with $i + 1 < j$. □

It is clear that this recurrence saves orthogonalizations, because one only has to orthogonalize against the previous two vectors of a subspace. Like the Arnoldi algorithm, the Nonsymmetric Lanczos algorithm has relations involving V_m, W_m, A, and A^T,

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T \tag{2.1}$$

$$AV_m = V_{m+1} \bar{T}_m \tag{2.2}$$

and

$$A^T W_m = W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T$$

$$A^T W_m = W_{m+1} \bar{T}_m^T$$

which yields

$$T = W_m^T A V_m$$

where V_m is an $n \times m$ matrix whose columns are the vectors $v_1, ..., v_m$, W_m is an $n \times m$ matrix whose columns are the vectors $w_1, ..., w_m$, $\bar{T}_m$ and $\bar{T}_m^T$ are the $(m+1) \times m$ tridiagonal matrices whose entries are obtained from the algorithm, and T_m and T_m^T are obtained from removing the $m + 1st$ rows of $\bar{T}_m$ and $\bar{T}_m^T$ respectively.

Nonsymmetric Lanczos generally has significant roundoff error that causes lose of biorthogonality. There can even be breakdown. The fix for these problems is the look-ahead method introduced by Parlett. [41]

2.4 Two-Sided Arnoldi

The two-sided Arnoldi algorithm is credited to Axel Ruhe [42]. This method has two subspaces, like nonsymmetric Lanczos, that are associated with A and A^T respectively. However the vectors, v and w, that are produced in this method are not biorthonormal.

The Two-Sided Arnoldi Algorithm

1. Perform the Arnoldi algorithm until right vector has converged, giving $AV_m = V_{m+1}\bar{H}_{mm}$.

2. Compute left eigenvector approximation:
$$t^T H_{mm} = z^T = t^T V_m^T$$

3. Perform the Arnoldi algorithm from the left

4. $w_1^T = z^T$

5. Compute w_j^T, orthonormal and K_{jj} lower Hessenberg, such that $W_j^T A = \bar{K_{jj}} W_{j+1}^T$.

6. Stop when left vector has converged indicated by
$$t^T K_{jj} = \nu t^T$$
$$|t_j k_{jj+1}| \ \ small$$

7. If possible, adjust vectors so that adjusted residuals $v\prime_{j+1}$ and $w\prime_{k+1}$ satisfy
$$w\prime_{k+1}^T V_j = 0$$
$$W_k^T v\prime_{j+1}.$$

2.5 Deflation in Krylov Methods

Techniques that remove eigenvalues from a problem are often called deflation techniques. A type of deflation can occur in Krylov type methods when approximate eigenvectors are used to augment a Krylov subspace. Deflation occurs automatically once the Krylov subspace grows large enough. However, in a restarted method a Krylov subspace might not grow large enough for this automatic deflation, and convergence will suffer accordingly. But for restarted methods, approximate eigenvectors can be calculated at the restart and kept for the next cycle. So when the approximate eigenvectors, called Ritz vectors, gain a certain accuracy, they "vanish" or deflate from the subspace. And this is what we define as deflation. It is known that deflation helps convergence in eigenvalue problems and when finding solutions to linear systems. In particular, deflation is most useful for eigenvalue problems where eigenvalues (small or large) are close together. For convergence of a particular eigenvalue, if there are approximate eigenvectiors in the subspace corresponding to nearby eigenvalues then they can be deflated and the particular eigenvalue converges faster. As for linear equations, convergence in Krylov subspace methods depends to a large degree on the distribution of eigenvalues. (Exceptions can be found for GMRES [14, 32].) Deflating or removing small eigenvalues helps increase the convergence rate.

2.5.1 Deflation in Restarted Methods

In this section, we look at some recent approaches that deflate restarted Krylov methods for both eigenvalue problems and linear equations. For eigenvalue computations with small matrices, deflation of eigenvalues that have completely converged is a natural part of the procedure [38]. The Rayleigh-Ritz procedure [38, 46] with large matrices can be assisted by other eigenvectors, even if they have not fully converged, in the computation of a particular eigenvector. One can notice the benefits of deflation even if these approximate eigenvectors are not very accurate. Convergence can improve, especially for problems with close eigenvalues, when several eigenvector approximations are kept in the subspace in non-Krylov Rayleigh-Ritz methods, such as Davidson's method [6, 19, 21, 22, 31, 57, 50, 58], and in subspace iteration [38, 46, 60, 16].

When restarting a Krylov method, the natural approach of beginning the next Krylov subspace with only one vector and this makes deflation impossible. Using block methods with addition storage and expense is a possible approach [46]. Much better is the implicity restarted Arnoldi method [56, 20, 24]. We first discuss simple restarted Arnoldi, then implicitly restarted Arnoldi and related methods.

2.5.2 Restarted Arnoldi

The non-restarted Arnoldi method can be rather expensive so restarting the method can reduce costs so Saad proposed a restarted Arnoldi method [43]. The idea behind the algorithm is to run the Arnoldi method for the first run. Then after the first run, you restart with a new starting vector for the second and subsequent runs. That new starting vector is a Ritz vector or a combination of two or more Ritz vectors. In the following example, we illustrate and explain the poor convergence of this method compared to non-restarted Arnoldi and nonsymmetric Lanczos.

Example 2.1. We compare the Arnoldi, nonsymmetric Lanczos, and restarted Arnoldi algorithms using a nonsymmetric, bidiagonal matrix of size 2000 with .1,1,2,...,1999 on the diagonal and 1's on the super diagonal. For restarted Arnoldi, we use the Ritz vector associated with the desired Ritz value for the starting vector at each restart. This residual plot is of the smallest eigenvalue, .1, and we can see in Figure 2.1 that the Arnoldi and Lanczos algorithms have similar convergence through 350 iterations while the restarted Arnoldi algorithm lags behind. In fact, it will take

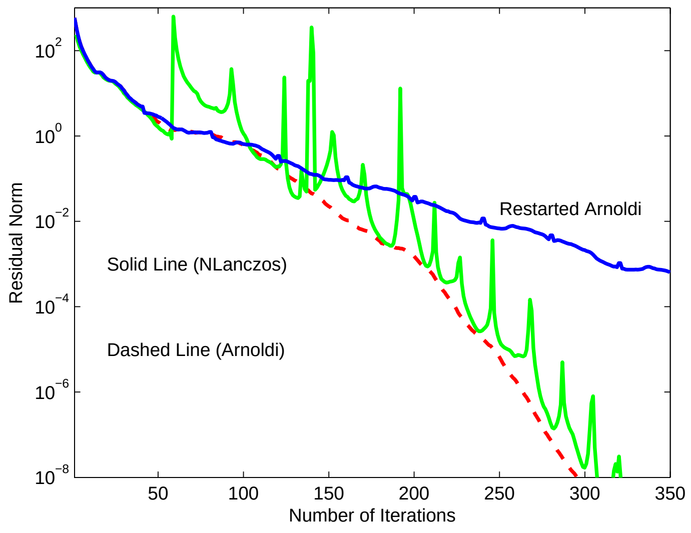

Figure 2.1: Comparison of Restarted Arnoldi with unrestarted Arnoldi and unrestarted nonsymmetric Lanczos.

restarted Arnoldi more than twice the number of iterations to converge compared to the Arnoldi and nonsymmetric Lanczos methods. This poor convergence for the restarted Arnoldi algorithm is due to the use of Ritz vectors at the restart. This occurs because the poor accuracy of the approximation. And when a combination of two or more Ritz vectors are used, the poorest approximation is transferred to all the approximations [24]. The "spiking" of the Lanczos convergence plot is typical and for Lanczos the same starting vectors are used, i.e. $v_1 = w_1$.

2.5.3 Implicitly Restarted Arnoldi

This method, due to Sorenson [56], is a way to restart the Arnoldi method while keeping all desired approximate eigenvectors and uses the subspace,

$$span\{y_1, y_2, ..., y_k, r, Ar, A^2r, ..., A^{m-k-1}r\}$$

where the $y_i's$ are the Ritz vectors computed at the end of and Arnoldi cycle and used in the next cycle, and r is a residual vector for one of the Ritz pairs. The subspace used in IRAM contains the subpaces,

$$span\{y_i, Ay_i, A^2y_i, ..., A^{m-k}y_i\}$$

where y_i is any of the k Ritz vectors saved from the previous cycle. So each Ritz vector has its own Krylov subapace that is contained in the larger Krylov subspace. So this a significant aid in the convergergenc of each Ritz vector respectively. This is due to the fact that $r_i = Ay_i - \theta_i y_i = \beta_i q_{m+1}$. So the vector q_{m+1} is a combination of Ay_i and y_i (for more see [24]).

2.5.4 Restarted Arnoldi with Eigenvector Approximations

The restarted Arnoldi with eigenvector approximations [24] uses the same Krylov subspaces as Sorenson's approach. Also this method is equivalent, at the end of a cycle, to the built-in MatLab function Eigs or ARPACK. It would be ideal for us to test against ARPACK. However, ARPACK factors often the matrix so this would not be a fair comparison. We use the Restarted Arnoldi with Eigenvector Approximations method instead of ARPACK when comparing the convergence of the eigenvalues with NLan-DR. Here we give the modern version of the restarted Arnoldi with eigenvector approximations that places the Ritz vectors at the beginning of the subspace [30].

Restarted Arnoldi with Eigenvector Approximations

1. *Start:* Choose m, the maximum size of the subspace, and k, the number of approximated eigenvectors that are retained from one cycle to the next. Also pick *numev*, the desired of eigenpairs. Specify σ, the target around which eigenvalues are desired. Choose an initial vector v_1 of unit length.

2. *Arnoldi iteration:* Apply the Arnoldi iteration from the current point to form the rest of V_{m+1} and $\bar{H}_m$. The current point is either v_1 if it is the first cycle or from v_{k+1} on the other cycles.

3. *Small eigenvalue problem:* Compute eigenpairs (θ_i, g_i), of H_m nearest σ.

4. *Check convergence* Residual norm can be computed using

$$\|Ay_i - \theta_i y_i\| = h_{m+1,m}|e_m^T g_i$$

   and convergence can be checked. If all desired eigenvalues have acceptable residual norm, then stop, first computing eigenvectors, if desired, as $y_i = V_m g_i$. Otherwise continue. The next step begins the restart.

5. *Orthonormalization of first k short vectors:* Orthonormalize g_i's, for $1 \leq i \leq k$, first separating into real and imaginary parts if complex, in order to form a real $m \times k$ matrix P_k. Both parts of complex vectors need to be included, so temporarily reduce k by 1 if necessary (or k can be increased by 1).

6. *Orthonormalization of the k+1 short vector:* Let $p_{k+1} = e^{k+1}$. This vector is already orthonormal.

7. *Form portions of new H and V using the old H and V* Let $\bar{H}_k^{new} = P_{k+1}^T \bar{H}_m P_k$ and $V_{k+1}^{new} = V_{m+1} P_{k+1}$. Then let $\bar{H}_k = \bar{H}_k^{new}$ and $V_{k+1} = V_{k+1}^{new}$. Converged eigenvectors can be locked in by zeroing out part of $\bar{H}_k$; see [30].

8. *Reorthogonalization of long k+1 vector:* Orthogonalize v_{k+1} against the earlier columns of the new V_{k+1}. Go to step 2.

2.5.5 Thick-restarted Lanczos (TRLAN)

This method is due to Wu and Simon and is used to solve symmetric eigenvalue problems. It is equivalent to implicitly restarted Lanczos (IRL) [3] where IRL is the special case of Sorenson's IRAM. The main advantage of this method is that it saves on orthogonalizations. Orthogonalizations are saved by using an $\omega-$ recurrence that lets you monitor orthogonality between the Lanczos vectors. So you do a partial orthogonalization but only when the $\omega-$ recurrence deems that orthogonalization necessary. This method also shows some effective restarting strategies [64].

2.5.6 Convergence Criteria for Eigenvalue Problems

- Want well relative separation between the eigenvalues.
- Want exterior eigenvalues.
- Don't want a starting vector that has small components in the direction of the eigenvectors that correspond to the eigenvalues you are interested in because initially convergence will be slow.

We will now switch from discussing the eigenvalue problem to the solution of the linear equations problem.

2.6 Methods for Solving Linear Equations

In this section, we discuss Krylov methods that solve systems of linear equations. But first, we give examples of how polynomials aid in the understanding of the convergence of these methods.

2.6.1 Convergence of Krylov Methods for Linear Equations

As stated earlier, polynomials aid in the understanding of how Krylov methods converge. We give the following theorem for convergence analysis in dealing with a system of linear equations. This theorem shows that the residual vector of a linear equations problem can be written in a polynomial.

Theorem 2.1. *Suppose that A has a basis of eigenvectors, then for the system of linear equations, $Ax = b$ the residual vector, r, can be written as $r = q(A)r_0$, where r_0 is the initial residual vector and q is a polynomial of degree m or less and $q(0) = 1$. Also r can be written as*

$$r = \sum_{i=1}^{n} q(\lambda_i)\beta_i z_i.$$

Proof. Let $\overline{x}$ be the initial guess for the solution to $Ax = b$. So the initial residual vector is $r_0 = b - A\overline{x}$ and we have

$$\begin{aligned} b - A\overline{x} &= r_0 \\ Ax - A\overline{x} &= r_0 \\ A(x - \overline{x}) &= r_0. \end{aligned}$$

The Krylov subspace associated with this is,

$$K_m(A, r_0) = Span\{r_0, Ar_0, A^2r_0, ..., A^{m-1}r_0\}.$$

Now pick, $\hat{x} \in K_m(A, r_0)$, where $\hat{x}$ is an approximate solution to the original system.

Then we have the residual vector

$$\begin{aligned} r &= b - A(\overline{x} + \hat{x}) \\ &= b - A\overline{x} - A\hat{x} \\ &= r_0 - A\hat{x}. \end{aligned}$$

Now we write $\hat{x}$ as,

$$\begin{aligned} \hat{x} &= c_1 r_0 + c_2 A r_0 + c_3 A^2 r_0 + ... + c_m A^{m-1} r_0 \\ &= (c_1 + c_2 A + c_3 A^2 + ... + c_m A^{m-1}) r_0 \\ &= p(A) r_0 \end{aligned}$$

where p is a polynomial of degree $m-1$ or less and defined as

$$p(\alpha) = c_1 + c_2\alpha + c_3\alpha^2 + ... + c_m\alpha^{m-1}.$$

Looking back at our residual vector r we have,

$$\begin{aligned} r &= r_0 - A\hat{x} \\ &= r_0 - Ap(A)r_0 \\ &= (I - Ap(A))r_0 \\ &= q(A)r_0 \end{aligned}$$

where $q(A) = I - Ap(A)$ and q is a polynomial of degree m or less and $q(0) = 1$. Let z_i's and λ_i's be eigenvectors and eigenvalues of A respectively. Now let $r_0 = \beta_1 z_1 + ... + \beta_n z_n = \sum_{i=1}^{n} \beta_i z_i$. So

$$\begin{aligned} q(A)r_0 &= q(A)(\sum_{i=1}^{n} \beta_i z_i) \\ &= \sum_{i=1}^{n} \beta_i q(A) z_i \\ &= \sum_{i=1}^{n} q(\lambda_i)\beta_i z_i \end{aligned}$$

because $Az_i = \lambda_i z_i$. $\square$

For this polynomial q we want it to be small over the spectrum of λ_i's.

2.6.2 How Polynomials Help Analyze Convergence

In this section, we give an example of how polynomials help analyze convergence. All the graphs use the same matrix that is bidiagonal with a diagonal of 0.1, 1,2,3,...,999, and a superdiagonal of all 1's.

Example 2.2. We begin with the GMRES (referred to in section 2.6.4) polynomial of degree 10. Recall, the ideal situation for convergence is for the value of the polynomial to be 1 at $x = 0$ and small over the eigenvalues.

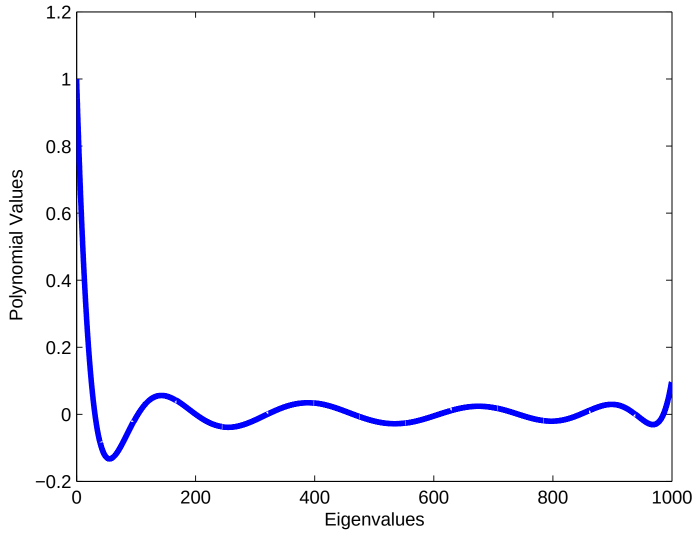

Figure 2.2. Behavior of a 10th degree GMRES polynomial over the eigenvalues

As seen in Figure 2.2, the polynomial is not small at the first eigenvalue of 0.1 and has this oscillating behavior over the remainder of the eigenvalues. This oscillating behavior is fine if the polynomial remains small over the eigenvalues, but this is not happening in this situation.

Now we take a broad and close look at the GMRES polynomial of degree 50. From Figure 2.3, the polynomial seems to be flattening out as the eigenvalues increase in size. However this polynomial is still not what we want. As we zoom in and take a closer look at this polynomial in Figure 2.4 , we can see that we still have a large polynomial value at the eigenvalue of 0.1. Also over the remaining eigenvalues, the

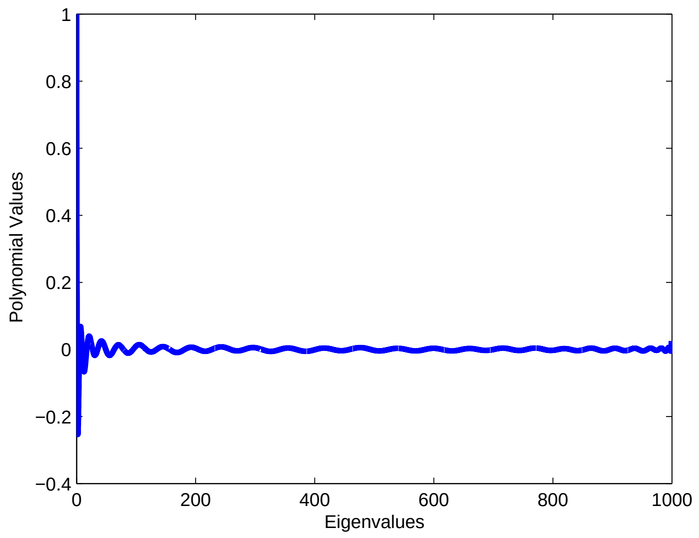

Figure 2.3. Behavior of a 50th degree GMRES polynomial over the eigenvalues

polynomial values are not small enough and there is still some oscillating behavior though it is getting better.

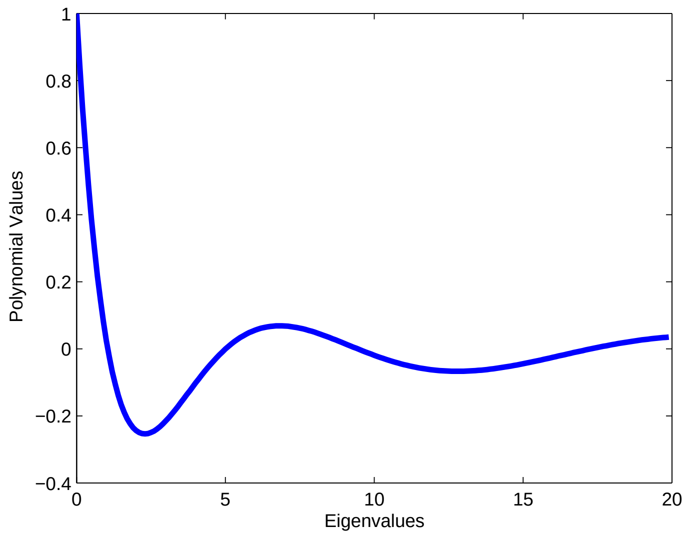

Figure 2.4: Behavior of a 50 degree GMRES polynomial over the eigenvalues (Close-Up View)

Figure 2.5 shows a polynomial of degree 100. We can see that for the eigenvalues of 2 and greater we are getting small polynomial values but the polynomial still is not small at the eigenvalues of 0.1 and 1.

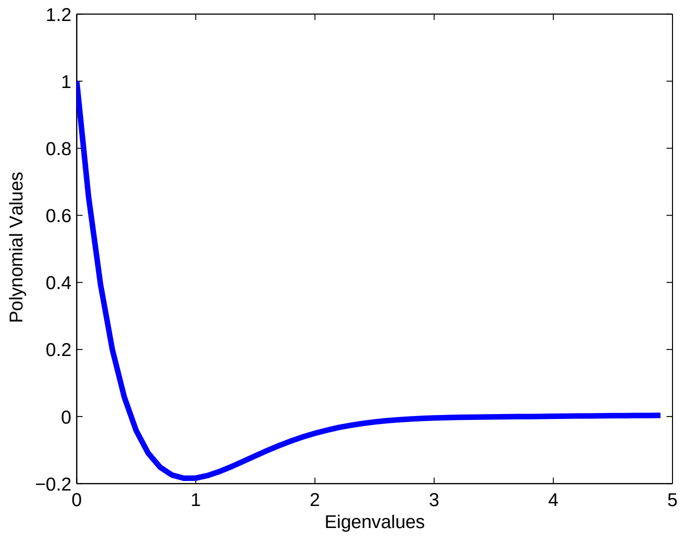

Figure 2.5. Behavior of a 100 degree GMRES polynomial over the eigenvalues

Figure 2.6 illustrates what we want. The polynomial is small over all the eigenvalues. Notice the polynomial dips down and comes back up before getting to zero at the eigenvalue of 1. This behavior is fine because we don't care how the polynomial behaves between the eigenvalues. We just require that the polynomial be one at zero and small over the eigenvalues.

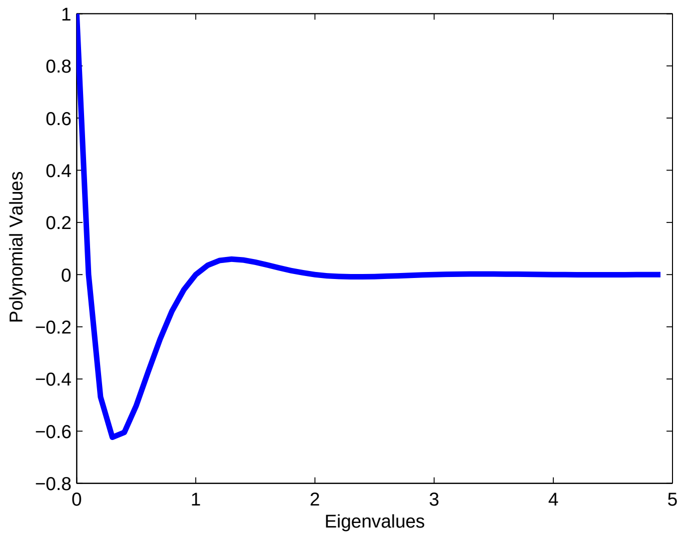

Figure 2.6. Behavior of a 150 degree GMRES polynomial over the eigenvalues

So we need a polynomial of degree 150 to get good convergence and this is illustrated in the next Figure 2.7. The figure shows the residual curve of the solution to the linear equations. We can see that between 100 and 150 matrix-vector products, which corresponds to the number of iterations in this case, that we are getting convergence of the linear equations.

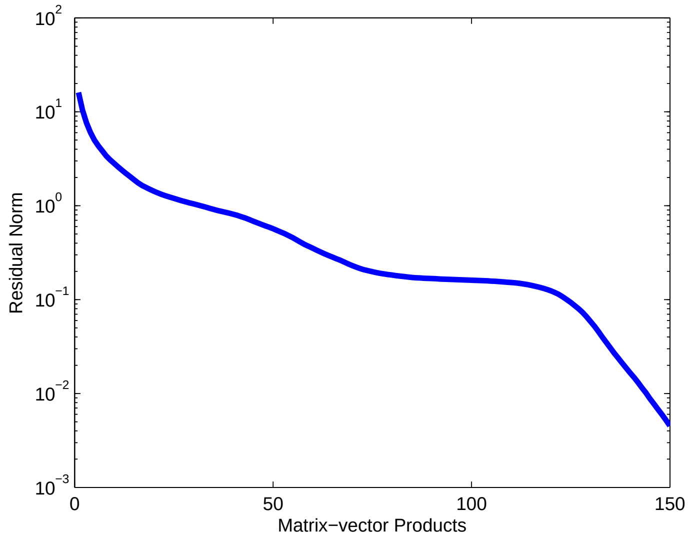

Figure 2.7: Convergence behavior of the Linear Equations problem associated with the polynomials

2.6.3 Conjugate Gradient (CG)

This method can be derived from the symmetric Lanczos algorithm and is best known for solving systems of sparse symmetric positive definite matrices. A symmetric positive definite matrix, call it A, is a symmetric matrix that has the property that for any nonzero vector $x \in R^n$, $x^T Ax > 0$. Equivalently, all of the eigenvalues are positive. From that, the CG method is not suitable for nonsymmetric systems of linear equations because the orthogonality of the residuals cannot be made with short recurrences [9, 63].

2.6.4 Generalized Minimum Residual (GMRES)

The generalized minimum residual method or GMRES is a more expensive method than the CG method due to the fact orthogonality of the residual vectors is maintained by using long recurrences. This requires more computation and causes a larger demand on storage. This method is a projection method over the Krylov subspace, K_m. The residual vector for GMRES has the following relation,

$$\begin{aligned} r = b - Ax &= b - A(\bar{x} + \hat{x}) \\ &= b - A\bar{x} - A\hat{x} \\ &= r_0 - A\hat{x} \\ &= \beta v_i - V_{m+1}\bar{H}_m d \\ &= V_{m+1}(\beta e_1 - \bar{H}_m d). \end{aligned}$$

So the residual norm is,

$$\begin{aligned} \|r\|_2 &= \|V_{m+1}(\beta e_1 - \bar{H}_m d)\|_2 \\ &= \|V_{m+1}\|_2 \|(\beta e_1 - \bar{H}_m d)\|_2 \\ &= \|(\beta e_1 - \bar{H}_m d)\|_2, \end{aligned}$$

since V_{m+1} is orthonormal. The GMRES is a byproduct of the Arnoldi algorithm. A stopping criteria for the GMRES method is to monitor the norm of the residual vector. One consequence of the breakdown (a division by zero in the algorithm) of the GMRES method is that if breakdown does occur then the algorithm will yield the exact solution. GMRES also has the same loss of orthogonality problems as the Arnoldi algorithm. So doing a double orthogonalization may help, but again that increases expense and storage. However to avoid these storage requirements and computational costs for orthogonalizing, GMRES is usually restarted. This method is called GMRES(m) because you restart after m iterations.

2.6.5 Biconjugate Gradient (BiCG)

Since the conjugate gradient method is not suited for nonsymmetric systems the biconjugate gradient method or BiCG was developed. As how the conjugate gradient method is associated with the symmetric Lanczos algorithm, the biconjugate gradient method is associated with the nonsymmetric Lanczos algorithm. BiCG is a projection method from,

$$K = span\{v_1, Av_1, A^2v_1, ..., A^{m-1}v_1\}$$

to

$$L = span\{w_1, A^Tw_1, (A^T)^2w_1, ..., (A^T)^{m-1}w_1\}.$$

Recall, the CG method cannot force the orthogonality of its residual vectors through short recurrences and GMRES method retains orthogonality of the residual vectors through an expensive long recurrence. However, the BiCG method can retain this orthogonality of residual vectors through a short recurrence, namely a three-term recurrence exactly like the nonsymmetric Lanczos method. In terms of number of iterations, BiCG is comparable to GMRES.

The convergence of BiCG is often irregular because BiCG is derived from the nonsymmetric Lanczos algorithm which also has irregular convergence behavior. Moreover, BiCG can experience breakdown problems like the nonsymmetric Lanczos algorithm and again these breakdown problems can be remedied through look-ahead procedures or by restarting at the iteration step before breakdown or near breakdown.

2.6.6 Quasi-Minimum Residual (QMR)

Recall that GMRES is a byproduct of the Arnoldi algorithm. This method, the Quasi-Minimum Residual method or QMR, is a byproduct of the nonsymmetric Lanczos algorithm. It is not possible to find the minimum residual solution with BiCG because the GMRES relation, $\|r\|_2 \neq \|\beta e_1 - \bar{T}_m d\|_2$, does not hold. However, $r = V_{m+1}(\beta e_1 - \bar{T}_m d)$ but when taking the norm of r it does not reduce like GMRES since V_{m+1} is not orthonormal. Nevertheless, QMR says minimize $\|r\|_2 = \|\beta e_1 - \bar{T}_m d\|_2$ anyway hence "quasi-minimum." The convergence behavior is smoother than BiCG. While QMR is not as fast as full GMRES and takes two matrix-vector products per iteration compared to GMRES, it can be faster than restarted GMRES. However like Lanczos, QMR can experience breakdown problems with look-ahead procedures or restarting are possible fixes though restarting will slow convergence.

2.6.7 Conjugate Gradient Squared (CGS)

This method is an alternative to BiCG and usually requires less iterations than BiCG. Where in BiCG A and A^T is used, the Conjugate Gradient Squared algorithm uses the matrix A twice. So this method is a welcome alternative when computing A^T is impractical. The residual norm with CG is $r = q(A)b$ while the residual norm with CGS is $r = q^2(A)b$. So the CGS method will converge twice as fast in number of iterations but not in terms of matrix-vector products. However if the convergence plot is not so "linear" then the CGS method is not as helpful compared to BiCG. Irregular convergence in the CGS method can have substantial build up of rounding errors. This leads to our next method.

2.6.8 Biconjugate Gradient Stabilized (BiCGStab)

The Biconjugate Gradient Stabilized method was developed to "smooth" the irregular convergence of the BiCG method and to remedy the roundoff problems of the CGS method, but it is more expensive than CGS and BiCG. Generally, BiCGStab converges about as fast as the CGS method however there are cases where it is faster and slower. One can think of the BiCGStab as a product of BiCG and a repeated application of GMRES. So from GMRES, BiCGStab minimizes the residual vector locally.

Example 2.3. In this example compare GMRES, BiCG, BiCGStab, and QMR using the same bidiagonal matrix that was used in Example 2.1. In Figure 2.8 we see that GMRES and BiCGStab are clearly the best of the four with GMRES being the best. It is well-known that GMRES cannot be beat in terms of matrix-vector products since it minimizes the residual. Also, the behavior of QMR and BiCG are expected since there are two matrix-vector products per iteration.

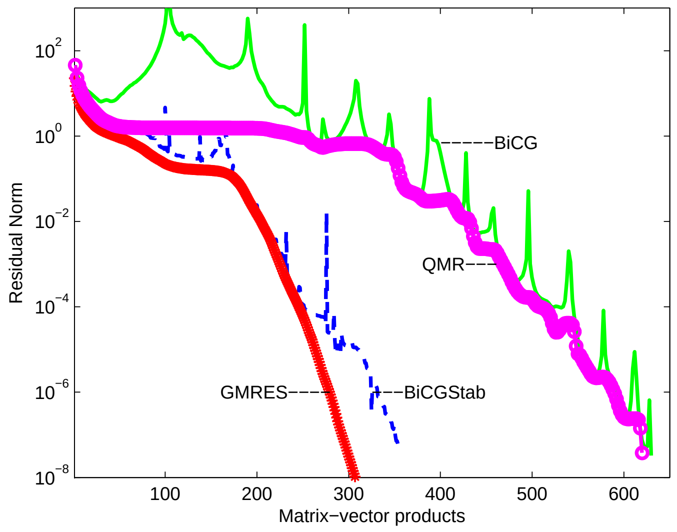

Figure 2.8. Comparison of a few linear equation methods

2.6.9 Transpose Free Quasi-Minimum Residual (TFQMR)

The best way to explain the TFQMR method,due to Freund, is to say it is the Quasi-Minimum Residual (QMR) method applied to CGS. However, QMR is linked to the nonsymmetric Lanczos algorithm which requires A^T. As stated in a previous section, CGS uses the matrix A twice rather than using A^T. Hence, the transpose free (TF) in TFQMR. It attempts to produce residual norms that are ”quasi optimal” in that an expression for residual norms is minimized by ”pretending” that the Lanczos vectors are orthonormal. Despite the lack of orthonormality, the residual vectors tend to behave smoothly and are monotone decreasing.

2.6.10 Harmonic Ritz Values

As an alternative to the regular Rayleigh-Ritz procedure, there are Harmonic Ritz approximations [21, 10, 36, 50, 27]. Given a subspace S and an orthonormal matrix V whose columns span S, regular Rayleigh-Ritz projects over the subspace using operator A. The small eigenvalue problem $V^TAVg = \theta g$ is solved. On the other hand, harmonic Rayleigh-Ritz projects over subspace $(A - \sigma I)S$ using the operator $(A - \sigma I)^{-1}$, where σ is a shift (possibly complex) in the region where eigenvalues are desired. The small eigenvalue problem becomes

$$V^T(A - \sigma I)^H(A - \sigma I)V\tilde{g} = (\tilde{\theta} - \sigma)(A - \sigma I)V\tilde{g}. \tag{2.3}$$

Harmonic Ritz pairs are $(\tilde{\theta}_i, \tilde{y}_i)$, where $\tilde{y}_i = V\tilde{g}_i$. From (2.3), Stewart [61] shows that if $\tilde{y}_i$ has unit norm, then

$$\|(A - \sigma I)\|\tilde{y}_i \leq |\tilde{\theta}_i - \sigma|. \tag{2.4}$$

So it is guaranteed that if $\tilde{\theta}_i$ is near σ, then the corresponding harmonic Ritz pair has a small residual. This means that harmonic Ritz pairs near σ are meaningful, even when σ is in the interior of the spectrum. This is not always the case for regular Ritz pairs.

The Rayleigh quotients of the harmonic Ritz vectors can be computed. We call these the harmonic Rayleigh quotients or ρ values. They often are more accurate than the harmonic Ritz values, particulary at early stages. However, the harmonic Rayleigh quotients do not have a property like the harmonic Ritz values do in (2.2), so they may not be as reliable. See [21, 26] for more on these ρ values.

2.6.11 GMRES with Deflated Restarting (GMRES-DR)

This method uses the thick restarting technique due to Wu and Simon [64]. As mentioned earlier, convergence for Krylov methods that solve linear systems depends greatly on the distribution of eigenvalues. This method is equivalent to the restarted method GMRES augmented with eigenvectors or GMRES-E at the end of each cycle. For more information on GMRES-E look to [23]. As for the algorithm the first cycle is regular GMRES, and the matrices H_m and V_{m+1} are produced from the Arnoldi iteration. The k smallest Ritz vectors are saved and orthonormalized after splitting them into real and imaginary parts. New forms of H_k and V_{k+1} are formed from old forms of H_m and V_{m+1}, then an Arnoldi iteration is performed to fill in the remaining parts of the new H_m and V_{m+1} respectively. Then an approximate solution,x_m, is formed, residual vector r is computed and Ritz pairs (θ_i, y_i) are computed. Again the k smallest Ritz vectors are saved (or others, if desired). Then restart with $x_0 = x_m$ and $r_0 = r$.

2.6.12 Convergence Criteria for Linear Equations

- Want eigenvalues to be well-separated from the origin (we don't want small eigenvalues)
- Want eigenvalues to not be too spread out
- Don't want negative eigenvalues or even worse eigenvalues spread on all sides of the origin.

2.6.13 Multiple Right-Hand Sides

Systems with multiple right-hand sides occur in many applications (see [11] for some examples). Block methods are a standard way to solve systems with multiple right-hand sides (see for example [34, 47, 11, 29, 15]). They put together Krylov subspaces for each right-hand side. If a non-restarted approach such as a block conjugate gradient method is used, then eventually there is quite a large space. This can give rapid convergence once eigenvectors corresponding to small eigenvalues are contained in this space. However, block methods are not ideal for every circumstance. They require all right-hand sides be available at the beginning. They have extra orthogonalization expense compared to non-block methods. They cannot always fully take advantage of related right-hand sides. Also, the implementation must be carefully done for stability (with removal or deflation of degenerate right-hand sides along the way).

Other approaches for multiple right-hand sides use information from the solution of the first right-hand side (and possibly others) to assist subsequent right-hand sides. Seed methods [55, 4, 18, 39, 45, 62, 8] project over entire subspaces generated while solving previous right-hand sides. Simoncini and Gallopoulos [53, 54] suggest methods including using blocks and using Richardson iteration with a polynomial generated from GMRES applied to the first right-hand side. In [28] a small subspace is generated with GMRES-DR applied to the first right-hand side that contains important information of approximate eigenvectors, and this is used to improve the subsequent right-hand sides. See [37] for a method for multiple right-hand sides that can also handle a changing matrix.

2.6.14 Problems in QCD

Many problems in lattice quantum chromodynamics (lattice QCD) have large complex systems of multiple right-hand sides. For example, the Wilson-Dirac formulation [12, 7] and overlap fermion [33, 1] computations both lead to such problems. Very large complex non-Hermitian matrices are needed. For Wilson-Dirac matrices, the right-hand sides represent different noise vectors.

CHAPTER THREE

Lanczos with Deflated Restarting (LAN-DR)

3.1 Introduction and Algorithm

We propose a restarted symmetric Lanczos method that both solves linear equations and computes eigenvalues and eigenvectors. It is called Lanczos with deflated restarting or Lan-DR. The Lan-DR method is a version of FOM-DR [25] for symmetric and Hermitian problems and is closely related to GMRES-DR [25]. As mentioned earlier, the eigenvalue portion of Lan-DR is TRLAN [64] and is mathematically equivalent to implicitly restarted Arnoldi (IRAM) [56]. For more on restarting symmetric Lanczos see [49].

For Lan-DR, the number of desired eigenvectors k must be chosen, along with which eigenvalues are to be targeted. Normally the eigenvalues nearest the origin are the most important ones for deflation purposes, but other eigenpairs can be computed. In particular, deflating large outstanding eigenvalues may help convergence of the linear equations solution and may be needed for stability.

At the time of a restart, let r_0 be the residual vector for the linear equations and let the Ritz vectors from the previous cycle be $\{y_1, y_2, \ldots, y_k\}$. Then the next cycle of Lan-DR builds the subspace

$$Span\{y_1, y_2, \ldots y_k, r_0, Ar_0, A^2r_0, A^3r_0 \ldots, A^{m-k-1}r_0\}. \tag{3.1}$$

Lan-DR generates the

$$AV_m = V_{m+1}\bar{T}_m, \tag{3.2}$$

where V_m is a n by m matrix whose columns span the subspace (3.1) and V_{m+1} is the same except for an extra column. Also $\bar{T}_m$ is an $m+1$ by m matrix that is upper-Hessenberg except for the $k+1$ by $k+1$ leading portion. This portion is non-zero only on the main diagonal (which has the Ritz values) and in the $k+1$ row and column. A part of recurrence (3.2) can be separated out to give

$$AV_k = V_{k+1}\bar{T}_k, \tag{3.3}$$

where V_k is an n by k matrix whose columns span the subspace of Ritz vectors, V_{k+1} is the same except for an extra column and $\bar{T}_k$ is the leading $k+1$ by k portion of T_m. This recurrence allows access to both the approximate eigenvectors (the Ritz vectors) and their products with A while requiring storage of only $k+1$ vectors of length n. The approximate eigenvectors in Lan-DR span a small Krylov subspace of dimension k.

It is necessary to maintain some degree of orthogonality of the columns of V_{m+1}. We suggest an approach to reorthogonalization that we call k-selective reorthogonalization (k-SO). For cycles after the first one, all new Lanczos vectors are reorthogonalized against the k Ritz vectors. This uses Parlett and Scott's idea of selective reorthogonalization [40], but is more natural in this setting, because we are already computing the approximate eigenvectors. Also, because of the restarting, there is no need to store a large subspace. Simon's partial reorthogonalization [52, 64] is also a possibility, as is periodic reorthogonalization [13].

Lan-DR is related to IRAM, GMRES-DR and TRLAN. However, it does something these other methods do not do. It efficiently solves systems of linear equations while simultaneously computing both eigenvectors and their related eigenvalues. We now give the algorithm for Lan-DR with k-selective orthogonalization.

The Lan-DR Algorithm

1. *Start.* Choose m, the maximum size of the subspace, k, the desired number of approximate eigenvectors. Set desired residual tolerance for eigenvalues, r_{tol}.

2. *First cycle and formation of the (k+1) by (k+1) portion of the new T.* Apply standard symmetric Lanczos Algorithm with starting vector v of norm 1 for

first cycle. This computes V_{m+1} and $\overline{T}_m$. In addition, fully reorthogonalize all v vectors of V_{m+1}. Compute the k smallest (or others, if desired) eigenpairs, (θ_i, g_i), T_m. Form T_{k+1}^{new}. T_{k+1}^{new} has zeros except on the main diagonal which has Ritz values, $\theta_1, ..., \theta_k$. The rest of T_{k+1}^{new} is formed element wise by $t_{k+1,i} = e_{m+1}^T T_{m+1,m} g_i$. Apply projection to solve first right hand side of linear equations. The projection is as follows, $V_m^T A V_m = V_m^T b \rightarrow T_m d = c$, where $V_m^T b = c$. Then solve for d and set $\hat{x} = V_m d$. From that, compute the residual norm $r = b - A\hat{x}$.

3. *Reassigning of the first k+1 vectors for second and subsequent cycles.* First form the Ritz vectors $y_i = V_{m+1} g_i$. Now for $i = 1, ..., k$ $v_i = y_i$. Also set $v_{k+1} = v_{m+1}^{old}$.

4. *Lanczos step from iteration k+2 to m of each cycle after cycle 1.* The remaining elements of $T_{m,m}$ are done using standard symmetric Lanczos from $k + 2nd$ iteration to the mth iteration. Using standard Gram-Schmidt, reorthogonalize against the new Lanczos vectors the first k vectors (k-SO). A full reorthogonalization of all m vectors from step 2 can be done if desired. Solve first right hand side of linear equations problem as in step 2.

5. *Computing k smallest Ritz pairs.* Compute the k smallest (or others, if desired) Ritz pairs (θ_i, g_i) of T_m. Form T_{k+1}^{new} in the same manner as in step 2.

6. *Computing residual norms.*

$$r_i = |\beta_{m+1}||g_{m,i}|\|v_{m+1}\|/\|y_i\|.$$

7. *Restart* Go to 3.

Full reorthogonalization can be used for the first cycle, but it may not be necessary. By Paige's theorem [35, 38, 61], orthogonality is only lost when some eigenvectors begin to converge. This may not happen in the first cycle if there are no outstanding eigenvalues.

We also suggest some hybrid approaches of k-SO. First, is k-PO (k partial orthogonalization). This approach uses Simons partial orthogonalization and you can use his ω-recurrence [52] to determine when you need to reorthogonalize. This approach cuts even more costs since we monitor the orthogonality from the ω-recurrence. The second hybrid approach is periodic k-SO. This is the same as k-SO but we only do it every so often. The form of the new T_m is exactly as it is for NLan-DR and will be discussed in detail in Chapter 4.

3.2 Convergence Theory

In this section, we give a theorem that concerns convergence theory of eigenvectors. This theorem deals with the case where the Ritz vectors are the actual eigenvectors. In particular this theory shows how deflation aids in convergence.

Theorem 3.1. *Assume that at a particular cycle of the Lan-DR algorithm the Ritz vectors from the previous cycle, $y_2, ..., y_k$, have attained an accuracy level so that they are the eigenvectors, $z_2, ..., z_k$, of the matrix A. Then those $k-1$ Ritz vectors can be deflated out in that we are able to extract a vector, y_1^{new}, from the Lan-DR subspace that has no components in the direction of $z_2, ..., z_k$.*

Proof. The Krylov subspace from the Lan-DR is the Krylov subspace augmented with Ritz vectors,

$$K = \{y_1, y_2, ..., y_k, v_{m+1}, Av_{m+1}, A^2v_{m+1}, ..., A^{m-k-1}v_{m+1}\},$$

where v_{m+1} is the residual vector. Also, $v_{m+1}^{old} = v_{k+1}$ where *old* means from the previous cycle. So assume that,

$$\{y_2, ..., y_k\} = \{z_2, ..., z_k\}$$

and let $y_1^{new} \in K$. Thus our new subspace is,

$$K = \{y_1, z_2, ..., z_k, v_{m+1}, Av_{m+1}, A^2 v_{m+1}, ..., A^{m-k-1} v_{m+1}\},$$

and $y_1^{new} = p(A)v_{m+1} + \sum_{i=2}^{k} \alpha_i z_i + \alpha_1 y_1$ where p is a polynomial of degree $m-k-1$ or less. Now we expand v_{m+1} in terms of the eigenvectors. So,

$$v_{m+1} = \sum_{i=1}^{n} \beta_i z_i.$$

Substituting this into the equation for y_1^{new} and we have,

$$\begin{aligned} y_1^{new} &= p(A)(\sum_{i=1}^{n} \beta_i z_i) + \sum_{i=2}^{k} \alpha_i z_i + \alpha_1 y_1 \\ &= \sum_{i=1}^{n} p(\lambda_i)\beta_i z_i + \sum_{i=2}^{k} \alpha_i z_i + \alpha_1 y_1 \\ &= \sum_{i=2}^{k} (\beta_i p(\lambda_i) + \alpha_i) z_i + \sum_{i=k+1}^{n} \beta_i p(\lambda_i) z_i + p(\lambda_1)\beta_1 z_1 + \alpha_1 y_1. \end{aligned}$$

Now let $\alpha_i = -\beta_i p(\lambda_i)$ for $i = 2, ..., k$ and from that we get that

$$y_1^{new} = \sum_{i=k+1}^{n} \beta_i p(\lambda_i) z_i + p(\lambda_1)\beta_1 z_1 + \alpha_1 z_1.$$

□

From this result, we can choose a polynomial, p, that has the value of 1 at λ_1 and needs to be small over the eigenvalues $\lambda_{k+1}, ..., \lambda_n$. So we can use a standard Tchebyshev polynomial to give concrete bounds on the accuracy of the vector y_1^{new} as an approximation to z_1.

3.3 Multiple Right-Hand Sides

Solution of systems with multiple right-hand sides is considered. We suggest a simple approach that uses the eigenvectors generated during the solution of the first right-hand side to deflate eigenvalues from the solution of the second right-hand side. First a projection is done over the Ritz vectors at the end of Lan-DR for the first right-hand side. Then the standard conjugate gradient method is used. We call this approach deflated CG or D-CG, since it deflates out eigenvalues before applying CG. It is similar to the init-CG approach [8] that also does a projection before CG.

3.4 Examples

Example 3.1. We use a test matrix that has many small eigenvalues. It is a diagonal matrix of dimension $n = 5000$ whose diagonal elements are $0.1, 0.2, 0.3, \ldots 9.8, 9.9, 10, 11, 12, \ldots, 4909, 4910$. The right-hand side is a random normal vector. We apply the method Lan-DR(100,40), which at each restart keeps the 40 Ritz vectors corresponding to the smallest Ritz values and then builds a subspace of dimension 100 (including the 40 approximate eigenvectors). With k-SO, at every iteration of all of the cycles except the first, there is a reorthogonalization against 40 vectors. We first look at how well Lan-DR computes the eigenvalues.

Figure 3.1 shows the residual norms for the 40 Ritz vectors. The desired number of eigenvalues is 30 and the desired residual tolerance is 10^{-8}. It takes 57 cycles for the first 30 eigenvalues to reach this level. Since this is a fairly difficult problem with eigenvalues clustered together, it takes a while for eigenvalues to start converging. However, from that point, eigenvalues converge regularly, and it can be seen that many eigenvalues and their eigenvectors can be computed accurately. The orthogonalization costs are significantly less than for fully reorthogonalized IRAM (about 84 vector operations for orthogonalization per Lan-DR iteration versus an average of 280 for IRAM). For a matrix that is fairly sparse so that the matrix-vector product is inexpensive (and also with cheap preconditioner, if there is one), the difference in orthogonalization is significant.

We continue the example by comparing Lan-DR(100,40) to unrestarted Lanczos. Figure 3.2 has the residual norms for the smallest and 30th eigenvalues with each method. The results are very similar for the first eigenvalue, in spite of the

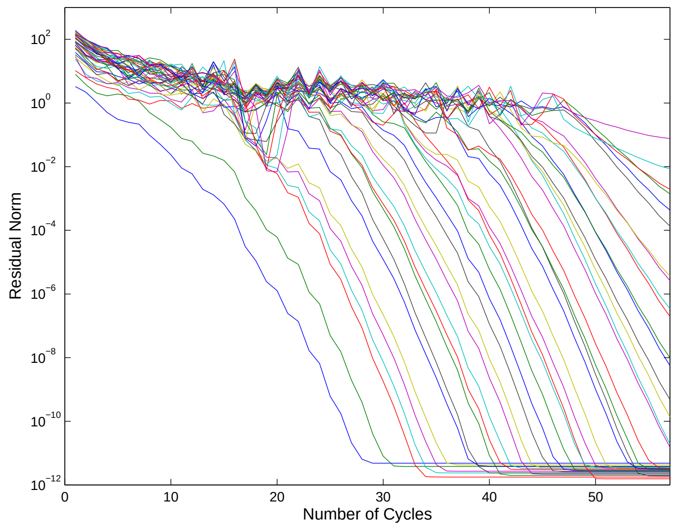

Figure 3.1. Computing many eigenvalues of a matrix with small eigenvalues.

fact that Lan-DR is restarted. The presence of the approximate eigenvectors corresponding to the nearby eigenvalues essentially deflates them and thus gives good convergence for Lan-DR. For eigenvalue 30, Lan-DR trails unrestarted Lanczos, but is still competitive. This is significant, since Lan-DR(100,40) requires storage of only about 100 vectors compared to nearly 3000 for unrestarted Lanczos.

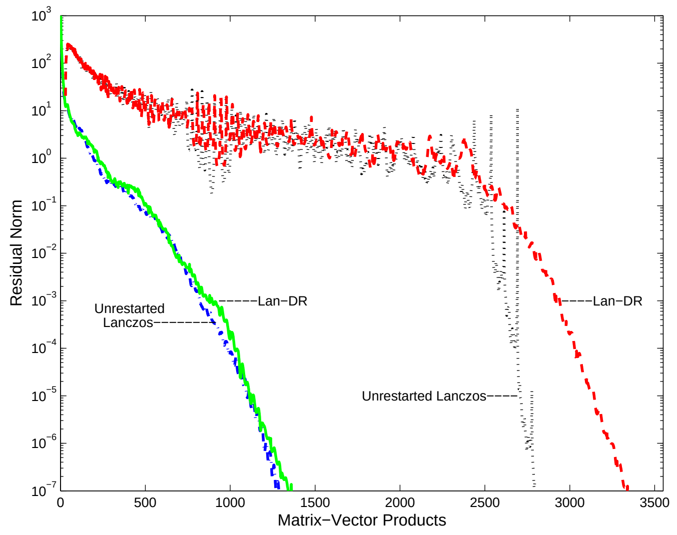

Figure 3.2: Comparison of Lan-DR(100,40) with unrestarted Lanczos for first and 30th eigenvalues.

Example 3.2. Next, we look at the solution of the linear equations. Figure 3.3 has Lan-DR with three choices of m and k and also has CG. The convergence of Lan-DR(100,40) is very close to that of CG. Lan-DR(100,0) has restarting every 100 iterations without saving any Ritz vectors, and it converges much slower. This is because only 100 steps of Lanczos does not generate very accurate approximate eigenvectors in its space. The deflation of eigenvalues in Lan-DR(100,40) allows it to compete with an unrestarted method such as CG. Lan-DR(100,10) is not too far behind CG, but Lan-DR(30,10) restarts too frequently and is much slower.

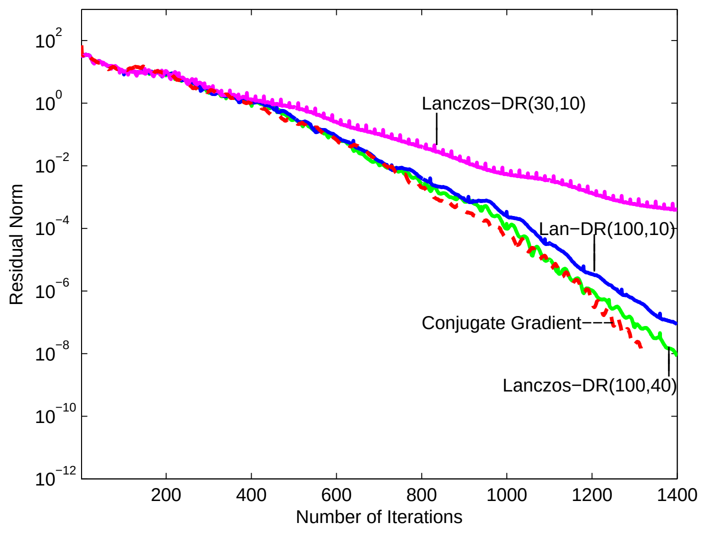

Figure 3.3. Comparison of Lan-DR with CG for solving linear equations.

Now we look at the vector operations for each of these methods. A vector operation is either a length n inner product or a length n daxpy (multiply a vector by a constant and add to another vector) or a similar operation. Lan-DR(100,40) performs 220348 vector operations while Lan-DR(100,10) performs 67036. Lan-DR(30,10) does not reach desired convergence but in 1410 iterations it performs 61920 vector operations. To that end, a comparison of vector operations of Lan-DR(30,10) and CG is not a fair one. CG only performs 3963 vector operations but this is due to two circumstances. The first is that CG ran less iterations to reach desired convergence than both Lan-DR(100,40) and Lan-DR(100,10) and second is that CG doesn't compute eigenvectors.

3.4.1 Reorthogonalization

It was mentioned earlier that some reorthogonalization is necessary to control roundoff error. We now look at this in more detail and give several possible approaches. The first is full reorthogonalization which takes every Lanczos vector formed by the three-term recurrence and reorthogonalizes it against every previous vector. The expense for this in terms of vector operations of length n varies from about $2k$ to $2m$ per iteration. This expense is significant if the matrix-vector product is fairly cheap, but may not be for a less sparse matrix with expensive matrix-vector product. The restarting of Lan-DR keeps the cost of full reorthogonalization down compared to non-restarted Lanczos. The next approach is periodic reorthogonalization [13]. For this, we always reorthogonalize the v_{k+1} and v_{k+2} vectors, then at regular intervals reorthogonalize two consecutive vectors (see [64, 61] for why consecutive vectors need to be reorthogonalized). The cost for this varies as with full reorthogonalization when it is applied, but it saves considerably if the reorthogonalization is not needed frequently. Next is partial reorthogonalization (PRO) [52, 51, 64, 61] which monitors loss of orthogonality and thus determines when to reorthogonalize. As suggested in [64], we use "global orthogonalization" against all previous vectors. As with periodic, we always reorthogonalize the v_{k+1} and v_{k+2} vectors and always reorthogonalize two consecutive vectors. This approach can be cheaper than periodic reorthogonalization, because it waits until reorthogonalization is needed.

The next three reorthogonalization methods are related to the three above, but reorthogonalization is done only against the first k vectors. Here we are using the idea from selective reorthogonalization (SO) [40] that orthogonality is only lost in the direction of converged or converging Ritz vectors [35, 38]. Since restarting is used, normally only the k Ritz vectors that are kept at the restart have an opportunity to converge. There can be exceptions as will be discussed in Examples 3.4 and 3.5 below. The first SO-type approach has been used in the previous section. It is called k-SO and has reorthogonalization at every iteration against the k Ritz vectors. This requires about $2k$ vector operations per iteration versus an average of about $k + m$ vector operations for full reorthogonalization. The last two methods are k-periodic and k-PRO. These are the same as periodic and PRO, except they reorthogonalize only against the first k vectors.

Example 3.3. We consider again Lan-DR for the matrix of Example 3.1. For this problem, loss of orthogonality is controlled by the restarting and the reorthogonalization of the v_{k+1} and v_{k+2} vectors at the restart. No further reorthogonalization is needed. Table 3.1 show a comparison with full reorthogonalization. The second column gives the loss of orthogonality as measured by $\|V_m^T V_m - I_{m\times m}\|$ at the end of 57 cycles. The next two columns have the residual norms of the first and thirtieth Ritz pairs. While full orthogonalization gives greater orthogonality of the Lanczos vectors, the Ritz vectors end up with similar accuracy. The thirtieth eigenvector continues to converge beyond cycle 57 and eventually reaches residual norm of 3.7×10^{-12} even with the approach of reorthogonalizing only at the restart. For this example, the Ritz vectors converge slowly enough that we don't have a Ritz vector appear and significantly converge in one cycle (see Figure 3.1). So before a eigenvector has converged, an approximation to it is among the group of Ritz vectors that v_{k+1} and v_{k+2} are reorthogonalized against. This explains why reorthogonalizing at restarts turns out to be often enough.

This example points out that restarting can make reorthogonalization easier. Reorthogonalization against only 40 vectors is done for two vectors every 60 iterations. If we compare to unrestarted Lanczos using PRO with global reorthogonalization and with tolerance on loss of orthogonality of square root of machine epsilon,

Table 3.1. Full reorthogonalization vs. reorthogonalize only at the restart

	orthogonality	rn 1	rn 30
k+1, k+2 vectors only	2.2×10^{-12}	5.6×10^{-12}	9.9×10^{-9}
full reorthog.	1.2×10^{-14}	6.7×10^{-12}	9.9×10^{-9}

the number of reorthogonalizations is similar. However, as the unrestarted Lanczos iteration proceeds, there are many previous vectors to reorthogonalize against. Also unrestarted Lanczos with PRO gives converged eigevectors with residual norms of just below 10^{-6} compared to well below 10^{-11} for Lan-DR(100,40) with reorthogonalization only at the restart. We note however, that the PRO tolerance can be adjusted for more frequent reorthogonalization and greater accuracy.

The next matrix is designed so that Lan-DR needs more reorthogonalization. We compare approaches and look at some potential problems.

Example 3.4. Let the matrix be diagonal with dimension $n = 5000$ and diagonal elements $1, 2, 3, \ldots 9, 10,\ 100, 101, 102, \ldots, 5088, 5089$. The right-hand side is a random normal vector. We use 10 cycles of Lan-DR(140,40) with the reorthogonalization approaches described at the beginning of this section. More reorthogonalization is needed than in Example 3.3, because eigenvectors converge quicker. Table 3.2 has the results. Two different tolerances on the loss of orthogonality for the PRO methods are used, square root of machine epsilon and three-quarters power. The second column of the table gives the frequency of reorthogonalization (of two vectors) for periodic methods. The last column has another measure of the effect of the roundoff error. It gives the number of iterations needed to solve a second right-hand side using the deflated CG method which will be given in the next section. We see that k-periodic reorthogonalizing of two vectors every 40 iterations is as good as k-SO with reorthogonalizing of every vector against the k Ritz vectors. Accuracy drops as the reorthogonalization is done less frequently. With frequency of 80, Lan-DR is still is able to compute some eigenvectors accurately, but unlike with frequency of 70, multiple copies of eigenvalues start appearing. Also, Lan-DR is not longer helpful for the solution of the second right-hand side.

Partial reorthogonalization gives results somewhat similar to periodic restarting without the need to select the frequency ahead of time. For k-PRO with $\epsilon_0^{.75}$, a total of 88 vectors are reorthogonalized compared to 60 for k-periodic with frequency of 40.

This example also demonstrates a problem that can happen when we reorthogonalize against only the k Ritz vectors. After the first cycle of Lan-DR with k-SO, there are only seven Ritz values below 10. After the second cycle, all ten small eigenvalues have converged to a high degree. Some orthogonality is lost in that cycle, because there are converged Ritz vectors in the subspace that are not reorthogonized against. This explains why the orthogonality of the basis is not as good for k-SO as for full reorthogonalization and also why other k-versions are not as good. It is dangerous to use k-SO if some eigenvectors converge very rapidly (within one cycle). The next example shows another possible problem with k-SO.

Example 3.5. For matrices with outstanding eigenvalues other than the small ones that can converge in one cycle, care must be taken. Ritz vectors corresponding to those eigenvalues need to be included in the Ritz vectors that are saved at the restart. We use the same matrix as in the previous example, except the largest eigenvalue is changed from 5089 to 5250. This eigenvalue is now outstanding enough to converge rapidly. Lan-DR with k-SO loses orthogonality of its basis. After 10 cycles, the orthogonality level is 7.2×10^{-3}, and it is actually worse (near 1) at earlier cycles. As mentioned, this can be fixed by including the large eigenpair among those selected to be saved for the next cycle.

Table 3.2. Compare reorthogonalization methods

	reor. freq.	orthogonality	rn 1	rn 30	2nd rhs it's
full	1	1.2×10^{-14}	7.0×10^{-12}	7.0×10^{-12}	57
k-SO	1	1.1×10^{-8}	7.0×10^{-12}	3.6×10^{-7}	57
k-periodic	40	8.4×10^{-10}	5.4×10^{-12}	2.7×10^{-8}	57
	60	1.8×10^{-7}	4.9×10^{-12}	6.1×10^{-7}	57
	70	4.0×10^{-6}	7.0×10^{-12}	2.5×10^{-5}	57
	77	1.4×10^{-1}	4.4×10^{-12}	1.0×10^{-4}	103
	80	2.0	5.3×10^{-12}	1.4×10^{-4}	210
periodic	70	4.2×10^{-6}	7.0×10^{-12}	2.6×10^{-5}	57
	80	3.3	5.3×10^{-12}	3.6×10^{-4}	212
k-PRO, $\epsilon_0^{.5}$	-	2.6×10^{-8}	5.3×10^{-12}	1.4×10^{-7}	57
k-PRO, $\epsilon_0^{.75}$	-	2.8×10^{-9}	4.4×10^{-12}	8.5×10^{-7}	57
PRO, $\epsilon_0^{.5}$	-	1.4×10^{-8}	5.3×10^{-12}	1.1×10^{-7}	57
PRO, $\epsilon_0^{.75}$	-	1.3×10^{-11}	4.4×10^{-12}	4.2×10^{-11}	57

3.4.2 Examples with Multiple Right-Hand Sides

Next, solution of systems with multiple right-hand sides is considered. We suggest a simple approach that uses the eigenvectors generated during the solution of the first right-hand side to deflate eigenvalues from the solution of the second right-hand side. First a projection is done over the Ritz vectors at the end of Lan-DR for the first right-hand side. Then the standard conjugate gradient method is used. We call this approach deflated CG or D-CG, since it deflates out eigenvalues before applying CG. It is similar to the init-CG approach [8] that also does a projection before CG.

Example 3.6. We consider the same matrix as in Example 3.1 and solution of a second right-hand side. The first right-hand side system has been solved with Lan-DR(100,40). We first illustrate how increasing the accuracy of the approximate eigenvectors helps the convergence of the second right hand side. Figure 3.4 has convergence curves for the second right-hand side with CG and with D-CG when Lan-DR has been run different numbers of cycles. D-CG always beats CG. However, D-CG is not as effective as it can be if Lan-DR has only been run 20 cycles. The eigenvectors are not all accurate enough to deflate out the eigencomponents from the residual of the second right-hand side, and eventually CG has to deal with these components and this slows convergence. So D-CG after 40 cycles converges rapidly

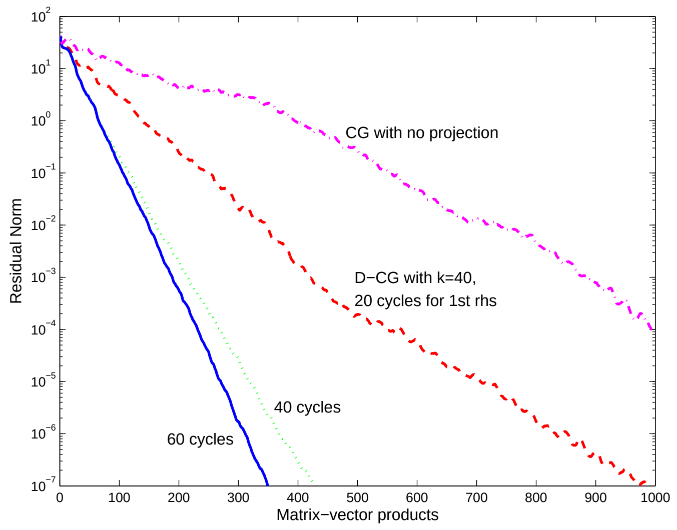

Figure 3.4. Solving the first right-hand side to different levels of accuracy.

and does not slow down as it procedes. Using 60 cycles of Lan-Dr is only a little better.

We next consider varying the number of approximate eigenvectors that are used for deflation. For the first right-hand side, Lan-DR(m,k) is run for 50 cycles with changing k and with $m = k + 60$. Figure 3.5 has the convergence results for applying D-CG to the second right-hand side. With $k = 10$ eigenvectors, D-CG is already significantly better than regular CG, but deflating even more eigenvalues is even better. For this example there is a significant jump upon going from 80 to 120 eigenvectors. This happens because having 120 gets past the 100 clustered eigenvalues pushes well into the rest of the spectrum.

Figure 3.6 shows that Lan-DR/D-CG can come out ahead of regular CG in terms of matrix-vector products even if we spend more on Lan-DR for the first right-hand side. We use 10 right-hand sides. The first is solved with 44 cycles of

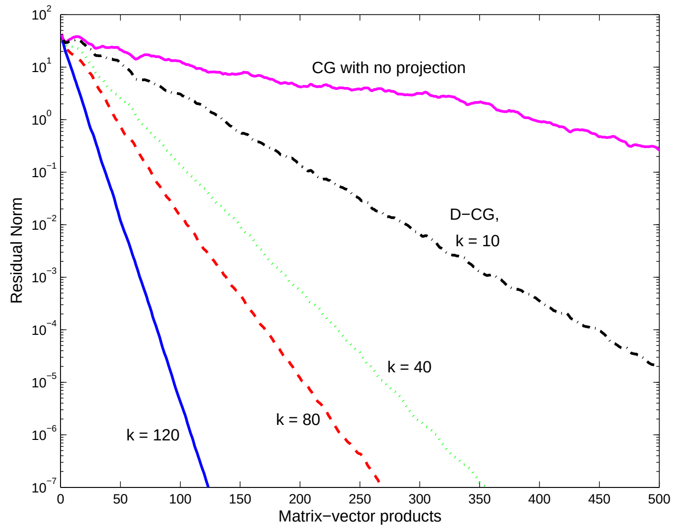

Figure 3.5. Deflating different numbers of eigenvalues.

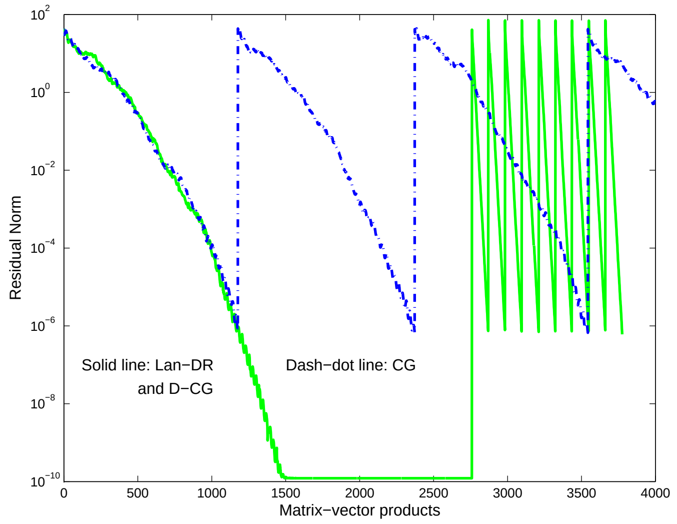

Figure 3.6. Lan-DR and D-CG versus regular CG for 10 right-hand sides.

Lan-DR(100,40). Then the other nine use runs of D-CG. This all takes about as many matrix-vector products as solving three systems with CG.

3.4.3 Comparison with Block-CG

Block-CG generates a Krylov subspace with each right-hand side as a starting vector then combines them all together into one large subspace. This large subspace can generally develop approximations to the eigenvectors corresponding to the small eigenvalues, so block-CG naturally deflates eigenvalues as it goes along. As a result, block-CG can be very efficient in terms of matrix-vector products.

Block methods require all the right-hand sides be available at the same time, while Lan-DR/D-CG only needs one right-hand side at a time. Simple block-CG can be unstable, particularly if the right-hand sides are related to each other. This can be controlled by the somewhat complicated process of removing (called “deflating”) right-hand sides.

Example 3.7. We let the matrix be diagonal with eigenvalues 1, 2, 3,...,10, 100, 101, 102,..., 5089. We compare the Lan-DR/D-CG approach with block-CG for 20 random right-hand sides. Figure 3.7 shows that the two approaches converge at almost the same number of matrix-vector products. However, Lan-DR has less orthogonalization expense.

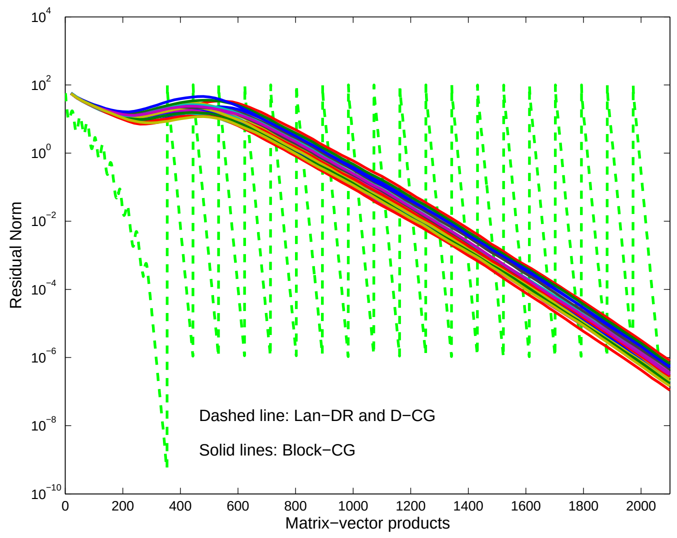

Figure 3.7. Comparison with Block-CG for 20 right-hand sides.

3.4.4 Related Right-Hand Sides

Example 3.8. We use the same matrix as in the previous example. There are again 20 right-hand sides, but this time the first is chosen randomly and the others are chosen as $b^{(i)} = b^{(1)} + 10^{-3} * ran^{(i)}$, where $ran^{(i)}$ is a random vector. The convergence tolerance is moved to relative residual below 10^{-6}, because block-CG has instability after that point. Figure 3.8 shows that Lan-DR does a better job of taking advantage of the related right-hand sides. As mentioned earlier, block-CG can be improved by removing right-hand sides once they become linear dependent.

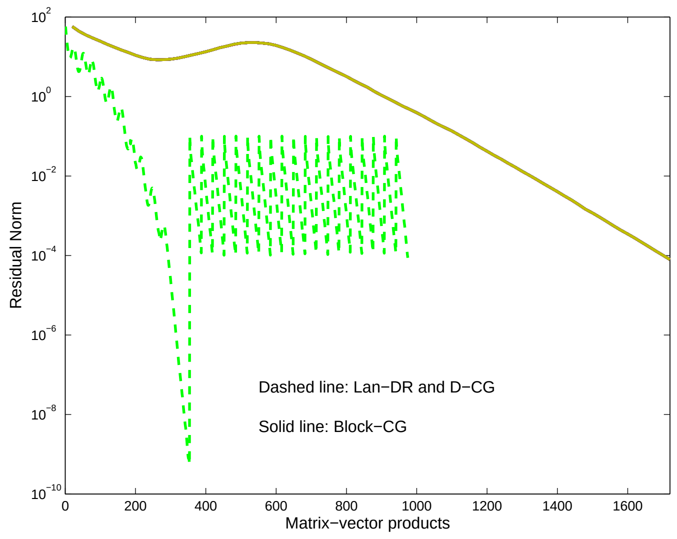

Figure 3.8. Comparison with Block-CG for 20 related right-hand sides.

3.4.5 Examples from QCD

Example 3.9. We choose a large QCD matrix of size $n = 1.5$ million. The *kappa* value is set near to *kappa*-critical, which makes it a difficult problem. There are generally a dozen or more right-hand sides for each matrix. The first test uses the $\gamma_5 M$ formulation which results in an indefinite matrix. The first right-hand side is solved with Lan-DR(m,k) with several values of k. Figure 3.9 shows the convergence for solution with the second right-hand side using either CG or D-CG with the different choices of k. We note that deflating 20 eigenvalues gives a big improvement over regular CG. Using 150 eigenvectors is almost an order of magnitude improvement over CG. The results in Figure 3.9 are fairly typical. Figure 3.10 shows results for the first five configurations (matrices) generated. D-CG with $k = 100$ eigenvalue deflated is compared with CG. There is some variance in CG, but with 100 small eigenvalues taken out, the convergence of D-CG is almost identical for all matrices.

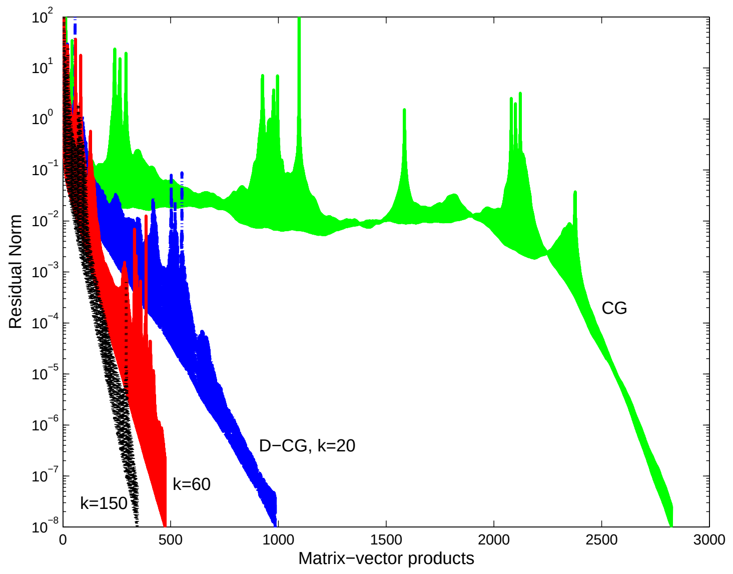

Figure 3.9: Comparison of CG to D-CG with varying numbers of eigenvalues deflated for a large QCD matrices.

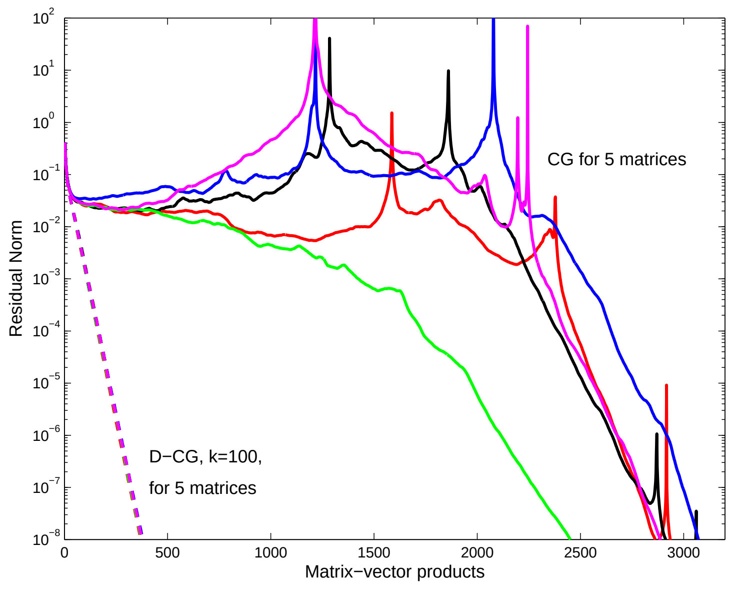

Figure 3.10: Comparison of CG to D-CG with k=100 eigenvalues deflated for five QCD matrics.

CHAPTER FOUR

Nonsymmetric Lanczos with Deflated Restarting (NLan-DR)

4.1 Introduction and Algorithm

It is known that while IRAM and restarted Arnoldi augmented with eigenvectors (simply called restarted Arnoldi from here on out) are reliable methods they can be expensive, and if one wants or needs to compute both left and right eigenvectors then two runs of IRAM or restarted Arnoldi are required. Hence, we offer the nonsymmetric Lanczos algorithm with deflated restarting (NLan-DR). For more on restarting the nonsymmetric Lanczos method see [59, 17]The benefits of deflation have already been discussed. So we focus on the facts that NLan-DR also is generally less expensive, from the three-term recurrence, and storage requirements are reduced compared to unrestarted Lanczos. Also, NLan-DR can compute both left and right eigenvectors simultaneously. Our application for needing both right and left eigenvectors is deflated BiCG-Stab. Options for computing left and right eigenvectors include running ARPACK twice, two-sided Arnoldi or nonsymmetric Lanczos. With nonsymmetric Lanczos there is limited storage and restarting may be necessary. NLan-DR is also equipped to solve systems of linear equations and in particular it can assist systems with multiple right-hand sides.

The NLan-DR Algorithm

1. *Start.* Choose m, the maximum size of the subspace, and k, the desired number of approximate eigenvectors.

2. *First cycle.* Apply standard Nonsymmetric Lanczos Algorithm. This computes V_{m+1}, W_{m+1} and $\overline{T}_m$. Compute the k smallest (or others, if desired) eigenpairs (θ_i, h_i), and (θ_i, g_i), left and right respectively, of T_m^T and T_m.

3. *Biorthonormalization of the first k vectors.* Orthonormalize the h_i's against the g_i's first separating into real and imaginary parts if complex in order to form m by k matrices H_k and G_k. (It may be necessary to adjust k in order to be certain both parts of the complex vectors are included.)

4. *Formation of a portion of new T using old T.* First extend the columns of H_k and G_k to length $m+1$ by appending a zero entry to each. Also add the $k+1st$ row, of length $m+1$, to both H_k and G_k all zeros except for placing 1 in the $m+1$ position. Now we have H_{k+1} and G_{k+1} are $m+1$ by $k+1$. Then let $T_{k+1} = H_{k+1}^T T^{old} G_{k+1}$.

5. *Reassigning of the first k vectors.* First form the approximate left and right eigenvectors, $u_i = W_{m+1} h_i$ and $y_i = V_{m+1} g_i$. Now for $i = 1, ..., k$ let $w_i = u_i$ and $v_i = y_i$.

6. *Computing residual norms.*

$$r_i^y = |\delta_{m+1}||g_{m,i}|\|v_{m+1}\|/\|y_i\|$$

$$r_i^u = |\beta_{m+1}||h_{m,i}|\|w_{m+1}\|/\|u_i\|$$

7. *Biorthonormalization of k+1 vector* Set $v_{k+1}^{new} = v_{m+1}^{old}$ and $w_{k+1}^{new} = w_{m+1}^{old}$. Form Av_{k+1} and $A^T w_{k+1}$ and biorthonormalize against the first $k+1$ vectors. This forms v_{k+2} and w_{k+2}.

8. *Lanczos Iteration* Apply nonsymmetric Lanczos from this point to form the remainder of T_{m+1}.

9. *Eigenvalue Computations.* Compute the k smallest (or others, if desired) left and right eigenpairs of T_m, (θ_i, h_i), and (θ_i, g_i).

10. *Restart.* Go to 3.

The Krylov subspaces that are produced during a run of NLAN-DR are,

$$Span\{y_1, y_2, ..., y_k, v_{m+1}, Av_{m+1}, A^2v_{m+1}, ..., A^{m-k-1}v_{m+1}\}$$
$$Span\{u_1, u_2, ..., u_k, w_{m+1}, A^T w_{m+1}, (A^T)^2 w_{m+1}, ..., (A^T)^{m-k-1} w_{m+1}\}$$

where the $y_i's$ are right Ritz vectors, the $u_i's$ are left Ritz vectors, and where v_{m+1} and w_{m+1} are the $m + 1st$ vectors from the previous cycle. But not only that, the left and right residual vectors, $r_i^u = A^T u_i - \theta_i u_i$ and $r_i^y = Ay_i - \theta_i y_i$, are multiples of the w_{m+1} and v_{m+1} vectors respectively.

We want to show that the vector spaces that are utilized by NLan-DR remain Krylov subspaces. Also, we only focus on the subspace associated with the right Ritz vectors because the proof for the left Ritz vectors is a very simple change to the proof of the right Ritz vectors. We have this from the algorithm:

$$Span\{v_1, v_2, ..., v_m\} = sp\{y_1, y_2, ..., y_k, v_{m+1}, Av_{m+1}, A^2v_{m+1}, ..., A^{m-k-1}v_{m+1}\}$$

where v_{m+1} is from the previous cycle. So we want to show that

$$Span\{y_1, y_2, ..., y_k, v_{m+1}, Av_{m+1}, A^2v_{m+1}, ..., A^{m-k-1}v_{m+1}\} = sp\{s, As, A^2s, ..., A^{m-1}s\},$$

for some vector s. We first proceed with the help of a lemma.

Lemma 4.1. *The residual vector for computing right eigenpairs of NLan-DR is a multiple of the v_{m+1} vector from the previous cycle i.e.*

$$r_i = Ay_i - \theta_i y_i = \gamma_i v_{m+1}.$$

Proof.

$$\begin{aligned} r_i &= Ay_i - \theta_i y_i \\ &= AV_m g_i - \theta_i V_m gi \\ &= V_m T_m - \theta_i V_m g_i + v_{m+1} t_{m+1,m} e_m^T g_i \\ &= V_m (T_m g_i - \theta_i g_i) + v_{m+1} t_{m+1,m} e_m^T g_i \\ &= v_{m+1} t_{m+1,m} e_m^T g_i \\ &= (t_{m+1,m})(g_{m,i}) v_{m+1} \\ &= \gamma_i v_{m+1} \end{aligned}$$

Line two of the proof uses the fact that $V_m g_i = y_i$, and line three uses Equation 2.2 and the fact that $\bar{T}_m = T_m + v_{m+1} t_{m+1,m} e_m^T$. A similar proof can be done for the left eigenpairs. □

Theorem 4.1. *We consider a cycle of NLan-DR. Let $r = v_{m+1}$ be the residual vector and $y_1, ..., y_k$ be the Ritz vectors calculated during the previous cycle. Then*

$$Span\{y_1, y_2, ..., y_k, v_{m+1}, Av_{m+1}, A^2 v_{m+1}, ..., A^{m-k-1} v_{m+1}\} = Span\{s, As, A^2 s, ..., A^{m-1} s\},$$

for some vector s. So the subspace for NLan-DR is a Krylov subspace.

Proof. From the lemma,

$$Ay_i - \theta_i y_i = \alpha_i v_{m+1} \tag{4.1}$$

First we claim that

$$Span\{y_1, ..., y_k\} = Span\{s, As, A^2 s, ..., A^{k-1} s\}, \tag{4.2}$$

for some vector s. Let

$$s = \sum_{i=1}^{k} \beta_i y_i. \tag{4.3}$$

Multiplying by A and using Eq. (4.1),

$$As = \sum_{i=1}^{k} \beta_i \theta_i y_i + \sum_{i=1}^{k} \beta_i \alpha_i v_{m+1}.$$

To have As in the span of the y_i vectors, we will make

$$\sum_{i=1}^{k} \alpha_i \beta_i = 0.$$

We set this equation up as part of a homogeneous system of linear equations with the β_i's as the unknowns. With v_{m+1} eliminated, we have

$$As = \sum_{i=1}^{k} \beta_i \theta_i y_i.$$

Multiplying again by A and using Eq. (4.1) gives

$$A^2 s = \sum_{i=1}^{k} \beta_i \theta_i^2 y_i + \sum_{i=1}^{k} \alpha_i \theta_i \beta_i v_{m+1}.$$

To again eliminate the v_{m+1} term, we let

$$\sum_{i=1}^{k} \alpha_i \theta_i \beta_i = 0.$$

So the next equation of our homogeneous system has the $\alpha_i \theta_i$'s as coefficients. Similarly, the next equation will have $\alpha_i \theta_i^2$'s as coefficients. We continue this until we have $A^{k-1}s$ is a linear combination of the y_i's. The homogeneous system then has $k-1$ equations. Since the system has k unknowns, there are linearly independent solutions for the β_i's. Putting these in (4.3) gives the desired vector s that satisfies Eq. (4.2). Again multiply by A and using Eq. (4.2) we have

$$A^k s = \sum_{i=1}^{k} \beta_i \theta_i^k y_i + \sum_{i=1}^{k} \alpha_i \theta_i^k \beta_i v_{m+1}.$$

So $A^k s$ is a linear combination of the y_i's and the v_{m+1}. $A^{k+1}s$ will be a linear combination of the y_i's, v_{m+1} and Av_{m+1}. We continue until we have $A^{m-1}s$ is a linear combination of the y_i's and v_{m+1} multiplied by the powers of A up to $m-k-1$. $\square$

The proof for the left Krylov subspace is the same with making the appropriate changes of A to A^T, y_i to u_i and v_{m+1} to w_{m+1}.

Recall, that the T_m matrix formed from the symmetric and nonsymmetric Lanczos algorithms is tridiagonal. However in NLan-DR (and Lan-DR) the T matrix has a varied form and we present that in the following section.

4.2 The Form of the new T for NLan-DR

We want to show that after restarting in the Lanczos Algorithm that the $m \times m$ matrix T has nonzero entries for $t_{i,i}$, $1 \leq i \leq m$, $t_{k+1,i}$, $1 \leq i \leq k$ and $k+1 < m$, and $t_{i,k+1}$, $1 \leq i \leq k$ and $k+1 < m$. Also T is tridiagonal below the $k+1st$ row.

Exactly like Lan-DR, the T_{m+1} matrix, after cycle 2, with $m = 10$ and $k = 5$ has the form,

$$T_{m+1} = \begin{pmatrix}
* & & & & & * & & & & & \\
& * & & & & * & & & & & \\
& & * & & & * & & & & & \\
& & & * & & * & & & & & \\
& & & & * & * & & & & & \\
* & * & * & * & * & * & * & & & & \\
& & & & & * & * & * & & & \\
& & & & & & * & * & * & & \\
& & & & & & & * & * & * & \\
& & & & & & & & * & * & * \\
& & & & & & & & & * & *
\end{pmatrix}$$

First we need the help of two lemmas to help us prove this new form of T.

Lemma 4.2. *W_m is orthogonal to $r_i = \frac{Ay_i - \theta_i y_i}{\|y_i\|}$, the right residual vector.*

Proof. For

$$r_i = \frac{Ay_i - \theta_i y_i}{\|y_i\|}$$

Since $\|y_i\| \neq 0$ we only need to show that W_m is orthogonal to the numerator of r_i. So we have,

$$\begin{aligned} W_m^T r_i &= W_m^T (Ay_i - \theta_i y_i) \\ &= W_m^T (AV_m g_i - \theta_i V_m g_i) \\ &= W_m^T A V_m g_i - \theta_i W_m^T V_m g_i \\ &= Tg_i - \theta_i g_i \\ &= 0. \end{aligned}$$

□

Lemma 4.3. *The v_{k+1} vector from the current cycle of NLan-DR is a multiple of v_{m+1} vector from the previous cycle or,*

$$v_{k+1} = v_{m+1}^{old}.$$

Similarly, $w_{k+1} = w_{m+1}^{old}$.

Proof. We have $span\{y_1, ..., y_k\} = span\{v_1, ..., v_k\}$ so $v_k = \sum_{i=1}^{k} y_i$. From this we have,

$$Av_k = A(y_k + \alpha_{k-1} y_{k-1} + ... + \alpha_1 y_1) \tag{4.4}$$

$$= \theta_k y_k + \theta_{k-1}\alpha_{k-1} y_{k-1} + ... + \theta_1 \alpha_1 y_1 + \gamma v_{m+1}^{old}. \tag{4.5}$$

Then v_{k+1} is formed by multiplying v_k by A and orthogonalizing against $v_1, ..., v_k$. So,

$$v_{k+1} = Av_k - \sum_{i=1}^{k} \beta_i v_i.$$

Now using Equation 4.4 and Equation 4.5 to substitute for Av_k and the fact that the $v_i's$ can be written as a combination of the $y_i's$ we have

$$\begin{aligned} v_{k+1} &= (\theta_k + \delta_k) y_k + \sum_{i=1}^{k-1} (\alpha_i \theta_i - \delta_i) y_i + \gamma v_{m+1}^{old} \\ &= \sum_{i=1}^{k} \eta_i v_i + \gamma v_{m+1}^{old} \\ &= \gamma v_{m+1}^{old}. \end{aligned}$$

The last line is true because $v_{k+1} \perp v_i$ for $1 \leq i \leq k$. So $\eta_i = v_i^T v_{k+1} = 0$. The proof for w_{k+1} is similar by making the $y_i's$ into $u_i's$, changing from right to left Ritz vectors, and changing $v's$ to $w's$. □

Proposition 4.1. *After restarting the nonsymmetric Lanczos Algorithm the $m \times m$ matrix T has nonzero entries for $t_{i,i}$, $1 \leq i \leq m$, $t_{k+1,i}$, $1 \leq i \leq k$ and $k+1 < m$, and $t_{i,k+1}$, $1 \leq i \leq k$ and $k+1 < m$. Also T is tridiagonal below the $k+1$st row.*

Proof. We prove this by selecting certain entries that are zero. Let $j < i < k$, so

$$\begin{aligned} t_{ij} &= w_i^T A v_j \\ &= w_i^T [A(y_j - \sum_{s=1}^{j-1} \alpha_s y_s)] \\ &= w_i^T [A y_j - \sum_{s=1}^{j-1} \alpha_s A y_s]. \end{aligned}$$

Now using the substitution $r_i^y = Ay_i - \theta_i y_i = \gamma_i v_{m+1}$ we have

$$\begin{aligned} & w_i^T [\gamma_j v_{m+1} + \theta_j y_j - \sum_{s=1}^{j-1} (\alpha_s \gamma_s v_{m+1} + \theta_s y_s)] \\ &= (\gamma_j - \sum_{s=1}^{j-1} \alpha_s \gamma_s) w_i^T v_{m+1} + w_i^T (\sum_{s=1}^{j} \theta_s y_s). \end{aligned}$$

The first dot product is zero due to Lemma 4.2 while the remaining dot products are equal to zero because w_i is orthogonal to each y_j. This is due to $sp\{v_1, ..., v_k\} = sp\{y_1, ..., y_k\}$. Now let $i < j < k$, so

$$\begin{aligned} t_{ij} &= w_i^T A v_j \\ &= (A^T w_i)^T v_j \\ &= [A^T (u_i - \sum_{s=1}^{i-1} \hat{\alpha}_s u_s)]^T v_j \\ &= [A^T u_i - \sum_{s=1}^{i-1} \hat{\alpha}_s A^T u_s)]^T v_j \\ &= [(r_i^u + \sum_{s=1}^{i-1} \hat{\alpha}_s r_s^u) + (\theta_i u_i - \sum_{s=1}^{i-1} \hat{\alpha}_s \theta_s u_s)]^T v_j \end{aligned}$$

Since $r_i^u = A^T u_i - \theta_i u_i = \hat{\gamma}_i w_{m+1}$ we have,

$$t_{ij} = (\hat{\gamma}_i + \sum_{s=1}^{i-1}) w_{m+1}^T v_j + (\theta_i u_i - \sum_{s=1}^{i-1} \theta_s u_s)^T v_j$$

The dot product involving w_{m+1} and v_j is zero because of the orthogonality of w_{m+1} to each v_j. The dot products involving the $u_i's$ and v_j because

$$sp\{w_1, ..., w_m\} = sp\{u_1, ..., u_k, A^T u_i, (A^T)^2 u_i, ..., (A^T)^{m-k-1} u_i\}.$$

To show $t_{k+1,i}$ is nonzero for $1 \leq i \leq k$, we have

$$\begin{aligned} t_{k+1,i} &= w_{k+1}^T A v_i \\ &= w_{k+1}^T (\gamma v_{m+1} + \sum_{s=1}^{k} \theta_s y_s) \\ &= \gamma w_{k+1}^T v_{m+1} + w_{k+1}^T (\sum_{s=1}^{k} \theta_s y_s). \end{aligned}$$

The dot products involving w_{k+1} and the $y_k's$ are all zero because w_{k+1} is orthogonal to each y_k. So that just leaves $\gamma w_{k+1}^T v_{m+1} = \alpha w_{m+1}^T v_{m+1}$ due to Lemma 4.3, and $\alpha w_{m+1}^T v_{m+1} = \alpha$. For $1 \leq i \leq k$,

$$\begin{aligned} t_{i,k+1} &= w_i^T A v_{k+1} \\ &= (A^T w_i)^T v_{k+1} \\ &= [A^T (u_i - \sum_{s=1}^{i-1} \hat{\alpha}_s u_s)]^T v_{k+1} \\ &= [A^T u_i - \sum_{s=1}^{i-1} \hat{\alpha}_s A^T u_s]^T v_{k+1} \\ &= (\gamma w_{m+1} - \sum_{s=1}^{i} \theta_s u_s)^T v_{k+1} \\ &= \gamma w_{m+1}^T v_{k+1}. \end{aligned}$$

And we know, by Lemma 4.3, $v_{k+1} = \delta v_{m+1}$, so we have

$$\begin{aligned} &\gamma w_{m+1} v_{k+1} \\ &= \beta w_{m+1} v_{m+1} \\ &= \beta, \end{aligned}$$

where $\beta = \gamma\delta$. From the $k + 2nd$ vector on, for both v_{k+1} and w_{k+1} in each cycle, NLAN-DR is the regular nonsymmetric Lanczos algorithm. Therefore T is tridiagonal from that point. □

4.3 Biconjugate Gradient with Deflated Restarting (BiCG-DR)

BiCG-DR is NLan-DR's linear equation solver. We give the algorithm for this as an addendum to NLan-DR's algorithm.

The BiCG-DR Algorithm

2.5 *Apply projection to the matrix A after the matrix T is formed.* After the formation of T_m a projection is applied to the linear equations problem $Ax = b$ and solve the small linear equations problem $T_m d = c$.

8.5 *Apply projection to the matrix A after the formation of T.* Apply same projection to A and follow the same process as in step 2.5. Then solve to desired tolerance.

Here we give more details of the projection process which is akin to the projection process for Lan-DR.

$$\begin{aligned} Ax &= b \\ W_m^T A V_m d &= W_m^T b \\ T_m d &= c \end{aligned}$$

where $T_m = W_m^T A V_m$ and $W_m^T = c$. Then solve for d and set $\hat{x} = V_m d$. From that we have,

$$\begin{aligned} A(x - \hat{x}) &= b - A\hat{x} \\ &= r, \end{aligned}$$

where r is the residual vector.

4.4 Deflated BiConjugate Gradient (D-BiCGStab)

To perform D-BiCGStab, for multiple right-hand sides (second and subsequent), first apply a projection using the right and left Ritz vectors from NLan-DR. This will be performed after the Ritz vectors have gained enough accuracy to deflate eigenvalues which means that the first right-hand side will usually run past a desired tolerance of 10^{-8}.

The D-BiCGStab Algorithm

1. *Apply projection on $Ax = b$ after the completion of NLan-DR algorithm.* For right hand sides 2 to number of right hand sides (nrhs), after the Ritz vectors have gained enough accuracy to deflate the eignvalues apply a projection on A using W_k and V_k to form the new linear equations problem $T_k d = c$.

Like BiCG-DR, we give further details of the projection process,

$$\begin{aligned} Ax &= b_{nrhs} \\ W_k^T A V_k d &= W_k^T b_{nrhs} \\ T_m d &= c \end{aligned}$$

where $T_k = W_k^T A V_k$ and $W_k^T = c$. Then solve for d and set $\hat{x} = V_k d$. From that we have,

$$\begin{aligned} A(x - \hat{x}) &= b_{nrhs} - A\hat{x} \\ &= r, \end{aligned}$$

where r is the right-hand side that is solved by BiCGStab.

4.5 Examples

Example 4.1. This first example is of NLan-DR against restarted Arnoldi using a bidiagonal matrix of size 2000 with $.1, 1, 2, \ldots.., 1999$ down the main diagonal and 1's along the super diagonal. The size of the subspace $m = 40$ and the desired number of eigenvalues $k = 10$ are the same for both NLan-DR and restarted Arnoldi. Figure 4.1 shows a comparison of convergence toward the smallest and sixth eigenvalues. NLan-DR and Restarted Arnoldi are nearly identical for the smallest and sixth eigenvalue. This version of NLan-DR uses the basic k-selective reorthogonalization scheme used in Lan-DR. So, we get similar results with NLan-DR with less cost. A cost comparison is discussed later.

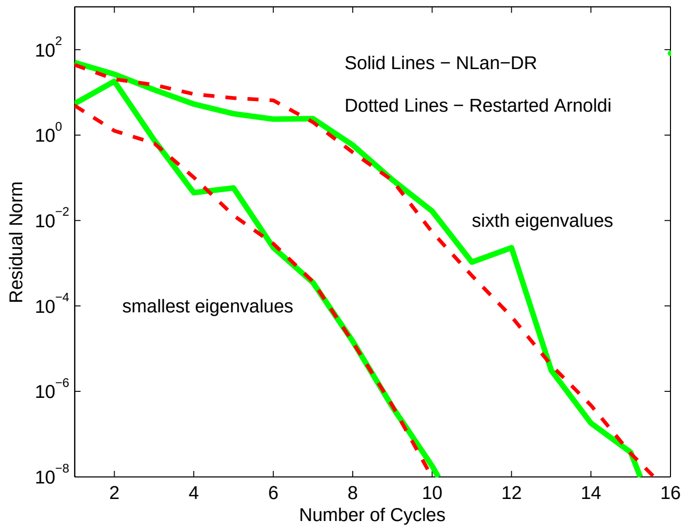

Figure 4.1: Eigenvalue comparison of NLan-DR and Restarted Arnoldi for a bidiagonal matrix of size 2000.

Example 4.2. In this second example the comparison of NLan-DR and restarted Arnoldi is given using the matrix Sherman4 from the Harwell-Boeing/Matrix Market collection. Sherman4 is an oil reservoir simulation matrix and is of size 1104. Again Figure 4.2 deals with the smallest and sixth eigenvalues, and NLan-DR uses the same rebiorthogonalization scheme as in the previous example. For the smallest eigenvalue, NLan-DR reaches a desired tolerance level of 10^{-8} in 10 cycles. Now for the sixth eigenvalue, we have near identical convergence like the previous example. Again we are seeing good convergence from NLan-DR compared to restarted Arnoldi

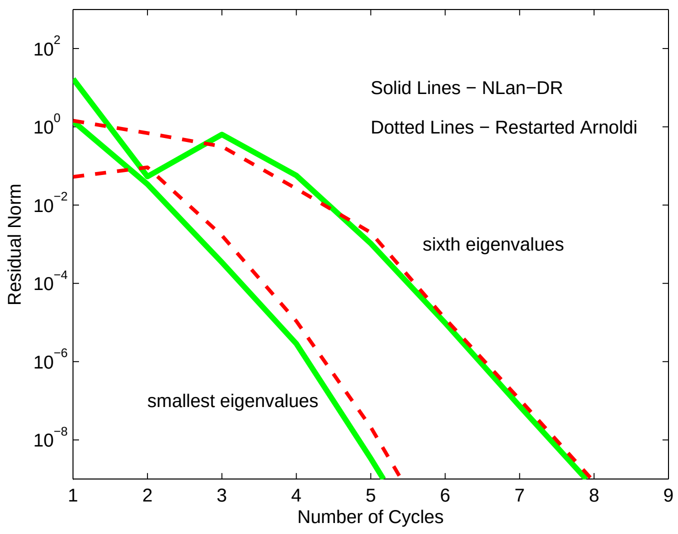

Figure 4.2: Eigenvalue comparison of NLan-DR and Restarted Arnoldi for the matrix Sherman4.

and also we know NLan-DR is a less expensive method than Arnoldi.

As stated earlier, the NLan-DR method is developed to also solve systems of linear equations (BiCG-DR) and is used to aid in deflating eigenvalues for BiCGStab (D-BiCGStab). Here we give a one example of each.

Example 4.3. For this example, we compare BiCG-DR with GMRES-DR. The matrix we use is the same bidiagonal matrix that is in example 4.1. Also we have the same size subspace of $m = 40$ and the same number of eigenvalues $k = 10$.

As seen in Figure 4.3, BiCG-DR is very competitive with GMRES-DR in number of iterations. BiCG-DR does require two matrix-vector products per cycle. This is justified if both left and right eigenvectors are desired. Also, we know that it is a less expensive method than GMRES-DR, because GMRES comes from the Arnoldi method.

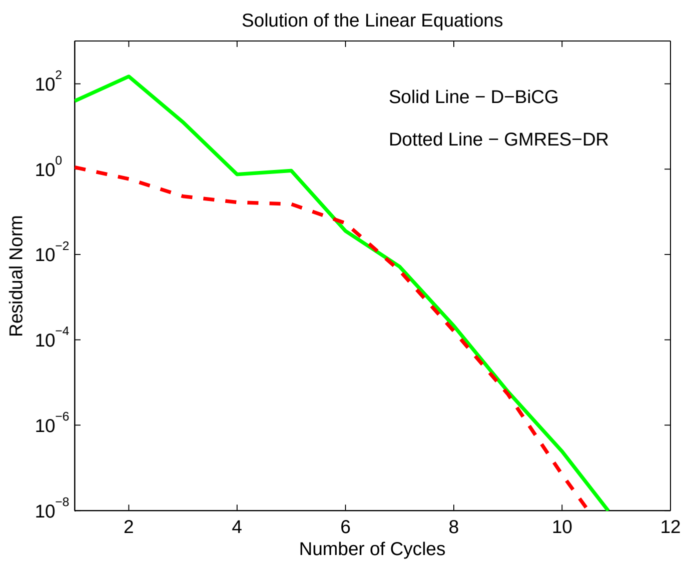

Figure 4.3: Linear equations comparison of NLan-DR and Restarted Arnoldi for a bidiagonal matrix of size 2000 of the first right hand side.

Example 4.4. In this example, we show the advantages of deflation for nonsymmetric linear systems with multiple right hand sides. This example uses the same matrix as in example 4.1 and has the same parameters of $m = 40$ and $k = 10$. So once the desired Ritz vectors have reach an accuracy that they will deflate, we see in Figure 4.4 that D-BiCGStab has converged in about 275 less matrix-vector products than non-deflated BiCGStab.

4.5.1 Expense

In this section we will compare the expense of NLAN-DR and restarted Arnoldi using size variations (n) of the matrix in Example 4.1 and we also increase the size of the subspace m as the size of the matrix increases. However, the number of desired eigenvalues $(k = 10)$ does not change. Vector operations (vops) are defined as they were in the previous chapter and matrix-vector products (mvps) are defined as a multiplication of the matrix A or A^T with an $n-$length vector. Also the the values for restarted Arnoldi are doubled to make a fair comparison for NLan-DR computing both left and right eigenvectors.

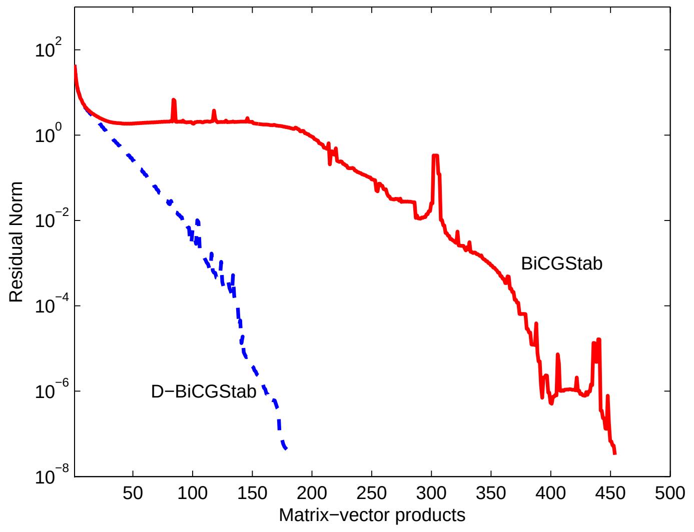

Figure 4.4: Comparison between non-deflated BiCGStab and D-BiCGStab for the second right hand side..

Table 4.1. Expense for Restarted Arnoldi and NLan-DR for k=10 eigenvalues

		Restarted Arnoldi			NLan-DR		
n	m	mvp	vops	cycles	mvp	vops	cycles
2000	40	980	67226	16	1089	31098	17
3000	55	1110	90090	12	1227	36393	13
5000	65	1560	142858	14	1600	47728	14
7000	75	1840	186258	14	2015	60167	15
10000	120	2000	289938	9	2248	82083	10

For these examples, NLan-DR again uses k-selective orthogonalization. Restarted Arnoldi uses no reorthogonalization due to it being a full orthogonalization method. As seen in Table 4.1, the ratios of vector operations for restarted Arnoldi to the vector operations for NLan-DR improves from 2.16 to 3.53 as we go from from the first example, $n = 2000$, to the last, $n = 10000$. NLan-DR does in most examples have more matrix-vector operations but this is due to going an extra cycle.

4.6 Roundoff Error

The nonsymmetric Lanczos algorithms have roundoff problems due to breakdown. A way to cure breakdown is the Look-Ahead Lanczos algorithm [41]. This algorithm essentially skips the breakdown step and continues from that point. However NLan-DR offers an alternative to the complicated Look-Ahead approach. Roundoff problems can also occur due to the loss of biorthogonality of the Lanczos vectors, v and w. We have seen examples where this is cured with selective reorthogonalization, partial reorthogonalization, or our k-selective reorthogonalization. In hard problems, our k-selective reorthogonalization is not enough to insure the biorthogonality of these vectors. So we have done a double k-selective reorthogonalization to treat some roundoff problems but also a full reorthogonalization is need for more severe roundoff problems. We also suspect that the ill-conditioning of the eigenvalues can cause roundoff problems and that is to be considered in future work.

CHAPTER FIVE

Two-Sided Arnoldi with Deflated Restarting (TSArn-DR)

5.1 Introduction and Algorithm

This method is given as an alternative to NLan-DR due to well-known instabilities of the Lanczos algorithm. We modify Ruhe's algorithm so that the Rayleigh Quotient matrix, here called G_m, is developed like the $T_m = W_m^T A V_m$ matrix in NLan-DR. Though more expensive, TSArn-DR does still produce both left and right eigenvectors. Like nonsymmetric Lanczos both a set of left and right vectors are developed, but instead of making them biorthogonal, we make them individually orthogonal. Also we feel that TSArn-DR is a better approach than running restarted Arnoldi twice, because you get both left and right eigenvectors simultaneously and if you do run restarted Arnoldi twice there is the opportunity for error when trying to match right and left eigenpairs.

The TSArn-DR Algorithm

1. *Start.* Choose m, the maximum size of the subspace, and k, the desired number of approximate eigenvectors.

2. *First cycle.* Apply standard Arnoldi Algorithm for the left and right sides where A^T is used for the left instead of A. This computes $V_{m+1} and W_{m+1}$. Form $G_m = W_m^T A V_m$ and $P_m = W_m^T V_m$. However in the formation of G_m use the Arnoldi recurrence for $AV_m = V_{m+1}\bar{H}_m$. So $G_m = W_m^T A V_m = W_m^T(V_{m+1}\bar{H}_m) = (W_m^T V_{m+1})\bar{H}_m = P_m H_m + W_m^T V_{m+1} h_{m+1,m}$. While running the separate left and right Arnoldi procedures, store the corresponding left and right Arnoldi matrices, T_m and H_m. These will be used later to compute left and right residual norms. Compute the k smallest (or others, if desired)

eigenpairs (θ_i, h_i), and (θ_i, g_i), left and right respectively, of the generalized eigenvalue problems $G_m g_i = \theta_i P_m g_i$ and $G_m^T h_i = \theta_i P_m^T h_i$.

3. *Compute residual norms.* Extend T_m and H_m to size $m+1$ by m by adding a row of zeros and in the $m+1, m$ entry, of both T_m and H_m, place the appropriate h_{j+1}, j. Now call the new matrices $\overline{T}_m$ and $\overline{H}_m$ respectively.

$$r_i^u = \|\overline{T}_m h_i - \hat{\theta}_i h_i\|$$

$$r_i^y = \|\overline{H}_m g_i - \theta_i g_i\|$$

4. *Orthonormalization of the first k vectors.* Orthonormalize the h_i's and the g_i's, first separating into real and imaginary parts if complex, in order to form m by k matrices $\hat{H}_k$ and $\hat{G}_k$. (It may be necessary to adjust k in order to be certain both parts of the complex vectors are included.)

5. *Formation of portions of new G, P, T, and H from old G, P, T, and H.*

$$G_k^{new} = \hat{H}_k^T G_m^{old} \hat{G}_k$$

$$P_k^{new} = \hat{H}_k^T P_m^{old} \hat{G}_k$$

$$T_k^{new} = \hat{H}_k^T T_m^{old} \hat{H}_k$$

$$H_k^{new} = \hat{G}_k^T H_m^{old} \hat{G}_k$$

6. *Reassigning of the first k vectors and forming the k+1st vectors.* First form the left and right eigenvectors, $u_i = W_m h_i$ and $y_i = V_m gi$. Now for $i = 1, ..., k$ let $w_i = u_i$ and $v_i = y_i$. Form the $k+1st$ vectors, $w_{k+1} = A^T w_k$ and $v_{k+1} = A v_k$ and reorthothogonalize these vectors against the previous k vectors. Form $Av_i = V_{m+1}^{old} \bar{H}_m g_i$ and $Aw_i = W_{m+1}^{old} \bar{T}_m h_i$.

7. *Reorthonormalization of the first k vectors and orthonormalization of the k+1st vectors.* Use full Gram-Schmidt to orthonormalize the first $k+1st$ left and right vectors.

8. *Forming the remainder of the matrices G, P, T, and H* For $k+1, ..., m$ apply Arnoldi iteration to both left and right sides as in cycle 1.

9. *Eigenvalue Computations.* Compute the k smallest (or others, if desired) left and right eigenpairs, (θ_i, h_i), and (θ_i, g_i).

10. *Restart.* Go to 3.

Both NLan-DR and TSArn-DR utilize the same Krylov subspaces of

$$span\{y_1, y_2, ..., y_k, r^y, Ar, A^2 r^y, ..., A^{m-k-1} r^y\}$$

$$span\{u_1, u_2, ..., u_k, r^u, A^T r^u, (A^T)^2 r^u, ..., (A^T)^{m-k-1} r^u\}$$

where the only difference is the residual vectors, r^y and r^u, for NLan-DR are the vectors, v_{m+1} and w_{m+1}, respectively from the previous cycle.

This explains the fact the convergence of NLan-DR and TSArn-DR are identical. Also we use TSArn-DR instead of running restarted Arnoldi twice because this simultaneously computes both left and right eigenvectors.

5.2 Examples

Example 5.1. Like the first example involving NLan-DR and restarted Arnoldi, we compare NLan-DR and TSArn-DR using the bidiagonal matrix of size 2000 with $.1, 1, 2, ..., 1999$ down the main diagonal and 1's along the super diagonal with $m = 40$ and $k = 10$. As in previous figures, Figure 5.1 shows a comparison of the first and sixth eigenvalues.

Example 5.2. Once again for our second example involving NLan-DR and TSArn-DR, we use the Sherman4 matrix with $m = 40$ and $k = 10$. Also the rebiorthogonalizations of NLan-DR are the same as the second example involving NLan-DR and restarted Arnoldi. Again from Figure 5.2 we see the identical convergence of the two methods.

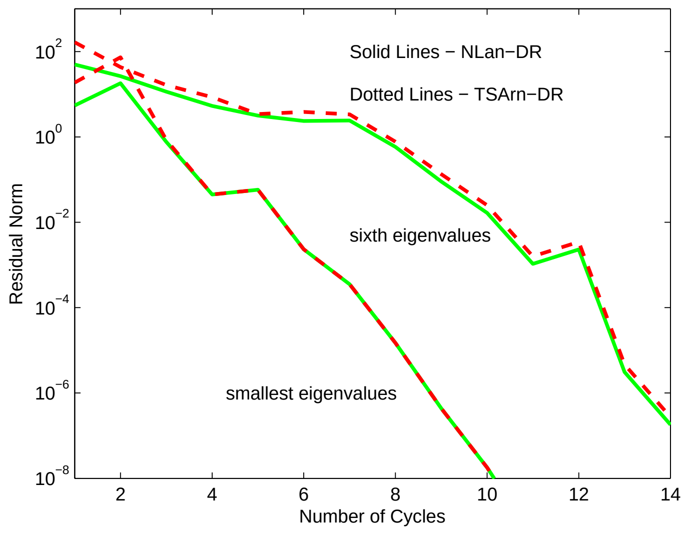

Figure 5.1: Eigenvalue comparison of NLan-DR and TSArn-DR for a bidiagonal matrix of size 2000.

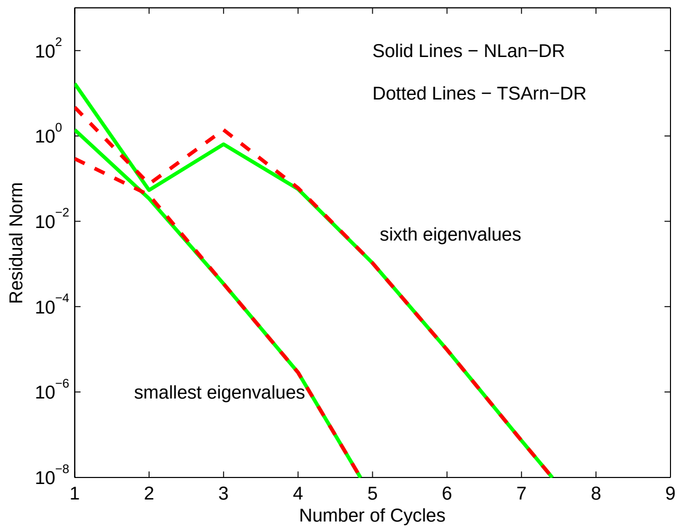

Figure 5.2: Eigenvalue comparison of NLan-DR and TSArn-DR for the matrix Sherman4.

CHAPTER SIX

Conclusions

We have given three new methods. The first two, Lan-DR and NLan-DR, find eigenvalues and simultaneously solve linear equations with multiple right hand sides. Both methods save on orthogonalizations compared to other methods so accurate results are found with less work. And while Lan-DR has been shown to be competitive with other methods in QCD problems and against block methods, NLan-DR needs further investigation. The third method, TSArn-DR, is given as an alternative to NLan-DR because of nonsymmetric Lanczos' instabilities. Both NLan-DR and TSArn-DR utilize the same subspace so they give identical results. We also feel TSArn-DR has an advantage over running the restarted Arnoldi method twice because TSArn-DR gives both left and right eigenvectors simultaneously.

BIBLIOGRAPHY

[1] G. Arnold, N. Cundy, J. van den Eshof, A. Frommer, S. Krieg, Th. Lippert, and K. Sch fer, *Numerical methods for the QCD overlap operator: II. Sign-function and error bounds,* 2003.

[2] W.E. Arnoldi. *The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem*, Quart. Appl. Math. 9 (1951), 17-29.

[3] D. Calvetti, L. Reichel, and D. C. Sorensen. *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*,Electron. Trans. Numer. Anal., 2 (1994), pp. 121.

[4] T.F. Chan and W.Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides,* SIAM J. Sci. Comput., 18 (1997), pp. 1698-1721.

[5] J.Daniel, W.B. Gragg, L.Kaufman, and G.W. Stewart. *Reorthogonlaization and Stable Algorithm for Updating the Gram-Schmidt QR Factorization*, Math. Comp. 30 (1976), 772-95.

[6] 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 (1975), pp. 87-94.

[7] D.Darnell, R.B. Morgan, and W.Wilcox, *Deflation of eigenvalues for iterative methods in lattice QCD,* Nucl. Phys. B (Proc. Suppl.), 129 (2004), pp. 856–858.

[8] J. Erhel and F. Guyomarc'h, *An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems,* SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1279-1299.

[9] V. Faber, and T. Manteuffel, *Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method.* SIAM J. Numer. Anal. 21, pp. 315-339, 1984.

[10] R.W. Freund, *Quasi-kernal polynomials and their use in non-Hermitian matrix iterations*, J.Comput. Appl. Math. 43 (1992) 135-138.

[11] R.W. Freund and M.Malhotra, *A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides,* Linear Algebra Appl., 254 (1997),pp. 119-157.

[12] A.Frommer, *Linear systems solvers - recent developments and implications for lattice computations,* Nucl. Phys. B (Proc. Suppl.), 53 (1997), pp. 120-126.

[13] J. Grcar, *Analyses of the Lanczos algorithm and of the approximation problem in Richardson's method,* PhD Thesis, University of Illinois at Urbana-Champaign, 1981.

[14] A. Greenbaum, V.Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465-469.

[15] M.H. Gutknecht. *Block krylov space methods for linear systems with multiple right-hand sides: an introduction, In A.H. Siddiqi, I.S. Duff, and O. Christensen, editors, Modern Mathematical Models, Methods and Algorithms for Real World Systems*, pp. 420-447. Anamaya Publishers, New Delhi, India, 2007.

[16] A. Jennings and W.J. Stewart, *A simultaneous iteration algorithm for real matrices*, ACM Trans. Math. Software, 7 (1981), pp. 184-198.

[17] E. Kokiopoulou, C. Bekas, E. Gallapoulos, *Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization,* Appl. Numer. Math., 49 (2004) 39-61.

[18] M.Kilmer, E.Miller, and C.Rappaport, *QMR-based projection techniques for the solution of non-Hermitian systems with multiple right-hand sides,* SIAM J. Sci. Comput., 23, pp. 761-78.

[19] 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 (1984), pp. 426-436.

[20] R.B. Lehoucq, *Analysis and Implementation of an Implicity Restarted Arnoldi Iteration*, Ph.D. thesis, Rice University, Houson, TX, 1995.

[21] R.B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl. 154-156 (1991) 289-309.

[22] R.B. Morgan, *Generalizations of Davidson's method for computing eigenvalues of large non-symmetric matrices*, J. Comput. Phys., 101 (1992), pp. 287-291.

[23] R.B. Morgan, *A restarted GMRES method augmented with eigenvectors,* SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154-1171.

[24] R.B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213-1230.

[25] R.B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20-37.

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

[27] R.B. Morgan, M. Zeng, *Harmonic projection methods for large non-symmetric eigenvalue problems*, Numer. Linear Algebra Appl. 5 (1998) 33-55.

[28] R.B. Morgan and W. Wilcox, *Deflated iterative methods for linear equations with multiple right-hand sides,* 2004.

[29] R.B. Morgan, *Restarted block-GMRES with deflation of eigenvalues,* Appl. Numer. Math., 54 (2005) pp. 222-236.

[30] R.B. Morgan, M.Zeng, *A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity*, Linear Algebra and its Applications, 415 (2006) 96-113.

[31] 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 (1992), pp. 382-389.

[32] N.M. Nachtigal, S.C. Reddy, and L.N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778-795.

[33] R. Narayanan and H. Neuberger, *An alternative to domain wall fermions,* Phys. Rev. D, 62:074504, 2000.

[34] D.P. O'Leary. *The block conjugate gradient algorithm and related methods,* Linear Algebra Appl., 29 (1980) pp.293-322.

[35] C.C. Paige, *The computation of eigenvectors and eigenvalues of very large sparse matrices,* PhD Thesis, University of London, 1971.

[36] C.C. Paige, B.N. Parlett, H.A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl. 2 (1995) 115-133.

[37] M.L. Parks, E. de Sturler, G. Mackey, D.D. Johnson, and S. Maiti, *Recycling Krylov subspaces for sequences of linear systems,* SIAM J. Sci. Comput., 28 (2006), pp. 1651-1674.

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

[39] B.N. Parlett, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations,* Linear Algebra Appl., 29 (1980), pp. 323-346.

[40] B.N. Parlett and D.S. Scott, *The Lanczos Algorithm with selective orthogonalization,* Math. Comp. 33 (1979), pp. 217-238

[41] B. Parlett, D. Taylor and Z. Liu, *A Look-Ahead Lanczos Algorithm for Unsymmetric Matrices,* Math. Comp. 44 (1985), pp. 105-124.

[42] A. Ruhe, *The Two-Sided Arnoldi Algorithm for Nonsymmetric Eigenvalue Problems*, Matrix Pencils, LMN 973, B. Kagstrom and A. Ruhe, eds., Springer-Verlag, Berlin Heidelberg New York, 1983, pp. 104-120.

[43] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices,* Linear Algebra Appl., 34 (1980), pp. 269-295.

[44] Y. Saad, *Krylov subspace methods for solving large unsymmetric linears systems.* Math. Comp. 37 (1981), pp. 105-126.

[45] Y.Saad,*On the Lanczos method for solving symmetric linear systems with several right-hand sides,* Math. Comp., 48 (1987), pp. 651-662.

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

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

[48] Y. Saad, *Iterative Methods for Sparse Linear Systems 2nd edition,* SIAM,Philadelphia, PA, 2003.

[49] Y. Saad, *Block Krylov-Schur method for large symmetric eigenvalue problems,* Report unmsi-2004-215, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2004.

[50] 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 (1996), pp. 401-425.

[51] H.D. Simon, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods,* Linear Algebra Appl., 61 (1984), pp. 101-132.

[52] H.D. Simon, *The Lanczos algorithm with partial reorthogonalization,* Math. Comp., 42 (1984), pp. 115-136.

[53] V.Simoncini and E.Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides,* SIAM J. Sci. Comput., 16 (1995), pp. 917-933.

[54] V.Simoncini and E.Gallopoulos, *A hybrid block GMRES method for nonsymmetric systems with multiple right-hand sides,* J. Comp. and Applied Math., 66 (1996), pp.457-469.

[55] C.Smith, A.Peterson, and R.Mittra. *A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields,* IEEE Trans. Anten. Propag, 37 (1989), pp.1490-1493.

[56] D.C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi Method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357-385.

[57] A. Stathopoulos, Y. Saad, and C. Fischer, *Robust preconditioning of large, sparse, symmetric eigenvalue problems*, J. Comput. Appl. Math., 64 (1995), pp. 197-215.

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

[59] A. Stathopoulos, *A case for a biorthogonal Jacobi-Davidson: restarting and correction equation*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 238-259.

[60] G.W. Stewart, *Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices*, Numer. Math., 25 (1976), pp. 123-136.

[61] G.W. Stewart, *Matix Algorithms II: Eigensystems*, SIAM, Philadelphia, 2001.

[62] H.A. van der Vorst, *An iterative method for solving f(A)x=b using Krylov subspace information obtained for the symmetric positive definite matrix A*, J. Comput. Appl. Math., 18 (1987), pp. 249-263.

[63] V. Voevodin, *The Problem of Non-Self-Adjoint Generalization of the Conjugate Gradient Method is Closed.* U.S.S.R. Comput. Maths. and Math. Phys. 23, 143-144, 1983.

[64] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602-616.