Polynomial filtered Lanczos iterations with applications in Density Functional Theory *

C. Bekas, E. Kokiopoulou, Yousef Saad
Computer Science & Engineering Dept.
University of Minnesota, Twin Cities.
{bekas,kokiopou,saad}@cs.umn.edu

July 15, 2005

Abstract

The most expensive part of all Electronic Structure Calculations based on Density Functional Theory, lies in the computation of an invariant subspace associated with some of the smallest eigenvalues of a discretized Hamiltonian operator. The dimension of this subspace typically depends on the total number of valence electrons in the system, and can easily reach hundreds or even thousands when large systems with many atoms are considered. At the same time, the discretization of Hamiltonians associated with large systems yields very large matrices, whether with plane-wave or real space discretizations. The combination of these two factors results in one of the most significant bottlenecks in Computational Materials Science. In this paper we show how to efficiently compute a large invariant subspace associated with the smallest eigenvalues of a Hermitian matrix using polynomially filtered Lanczos iterations. The proposed method does not try to extract individual eigenvalues and eigenvectors. Instead, it constructs an orthogonal basis of the invariant subspace by combining two main ingredients. The first is a filtering technique to dampen the undesirable contribution of the largest eigenvalues at each matrix-vector product in the Lanczos algorithm. This techniques employs a well-selected low pass filter polynomial, obtained via a conjugate residual-type algorithm in polynomial space. The second ingredient is the Lanczos algorithm with partial reorthogonalization. Experiments are reported to illustrate the efficiency of the proposed scheme compared to state-of-the-art implicitly restarted techniques.

Keywords: Polynomial Filtering, Conjugate Residual, Lanczos Algorithm, Density Functional Theory

1 Introduction

A number of scientific and engineering applications require the computation of a large number of eigenvectors associated with the smallest eigenvalues of a large symmetric/hermitian matrix. An important example is in electronic structure calculations where the charge density $\varrho(r)$ at a point r in space is calculated from the eigenfunctions Ψ_i of the Hamiltonian $\mathcal{A}$ via the formula

$$\varrho(r) = \sum_{i=1}^{n_o} |\Psi_i(r)|^2, \tag{1}$$

where the summation is taken over all occupied states (n_o) of the system under study. This is a crucial calculation in Density Functional Theory since the potential V of the Hamiltonian $\mathcal{A} = \nabla^2 + V$, depends

*Work supported by NSF grants NSF/ITR-0325218 and NSF/ITR-0428774, by DOE under Grants DE-FG02-03ER25585, DE-FG02-03ER15491, and by the Minnesota Supercomputing Institute

on the charge density ϱ, which in turn depends on the eigenvectors ψ_i of $\mathcal{A}$ (see (1)), and as a result, an iterative loop is required to achieve self-consistence. Computing the charge density $\varrho(r)$ via (1), requires eigenvectors, though it is more accurate to say that what is needed is an orthogonal basis of the invariant subspace associated with the n_o algebraically smallest eigenvalues. This is because $\varrho(r)$ is invariant under an orthogonal transformations of the basis of eigenfunctions $\{\Psi_i\}$. If the symmetric matrix A is the discretization of the Hamiltonian $\mathcal{A}$ and the vectors ψ_i are the corresponding discretizations of the eigenfunctions $\Psi_i(r)$ with respect to r, then, the charge densities are the diagonal entries of the "functional density matrix"

$$P = Q_{n_o} Q_{n_o}^\top \quad \text{with} \quad Q_{n_o} = [\psi_1, \ldots, \psi_{n_o}]. \tag{2}$$

Specifically, the charge density at the j-th point r_j is the j-th diagonal entry of P. In fact, any orthogonal basis $\mathcal{Q}$ which spans the same subspace as the eigenvectors ψ_i, $i = 1, \ldots, n_o$ can be used. This observation has lead to improved schemes which do not focus on extracting individual eigenvectors. For example, [4] showed that the semi-orthogonal basis computed by the Lanczos algorithm with partial reorthogonalization can be used in order to extract accurate approximations to the charge density. This scheme results in substantial savings relative to schemes which rely on the full reorthogonalization of the Lanczos vectors.

In simple terms, the problem considered in this paper can be stated as follows. Given a real symmetric (or complex Hermitian) matrix $A \in \mathbb{R}^{n\times n}$ with eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, compute the invariant subspace $\mathcal{S}_{n_o}$ associated with the eigenvalues which do not exceed a certain limit γ. In electronic structures, γ is the Fermi energy level and the interval $[a, \gamma]$ contains the (algebraically) smallest occupied eigenstates eigenvalues $\lambda_1, \ldots, \lambda_{n_o}$. We assume that we are given an interval $[\alpha,\ \beta]$ which (tightly) contains the spectrum of A. The nature of the algorithms used also requires that $\alpha \geq 0$. If this is not satisfied, we shift matrix A by a scalar σ so that $A + \sigma I$ does not have any negative eigenvalues. Methods for computing an interval $[\alpha,\ \beta]$ when this is not readily available are discussed in section 3.1.

This well known problem received much attention in the past few decades, as can be seen from the survey [2][1] which provides a good illustration of the wealth of algorithmic research in this field. For a comprehensive theoretical discussion of the problem one can to refer to Parlett's classic book [24] and the references therein.

When the number of desired eigenvalues is rather small, in the order of a few dozens, a variety of algorithms successfully address the problem. The most general purpose and extensively used method is based on implicitly restarted Lanczos iterations [36] and is implemented in the software package `ARPACK` [16]. However, the problem becomes particularly demanding in the cases when we seek to compute a large number of eigenvalues that reach deep into the interior of the spectrum of the matrix at hand. Indeed, in electronic structure calculations, the dimension of the corresponding invariant subspace is equal to the number of occupied states n_o which typically depends upon the number of free electrons of the system under study. Current state of the art calculations may involve hundreds or even thousands of states. In addition, the dimension n of the Hamiltonian A also depends on the number of atoms and the topology of the system and is almost always in the order of a few hundred thousands.

There is a rather rich variety of ways to compute bases of large eigenspaces. Of all methods, the subspace iteration algorithm is probably the simplest [24]. This algorithm computes the dominant eigenspace, i.e., the one associated with the largest modulus of A, by simply computing an orthogonal basis of the span of $A^k X$, where X is an $n \times p$ initial approximation to the egenbasis. This method can be easily modified to compute the eigenspace associated with the algebraically smallest eigenvalues by shifting the matrix. In fact, the subspace iteration algorithm is often used with Chebyshev acceleration [26] which replaces the projection subspace $A^k X$ by $C_k(\theta I + (A - \sigma I)/\eta)X$, where C_k is the Chebyshev polynomial of degree k of the first kind and θ, σ, η are scaling and shifting parameters. This method was one of the first to be used in electronic structure calculations [17, 18] and this might be attributed to its simplicity and to the availability of an old `ALGOL` procedure published in the Wilkinson and Reinsch collection of algorithms [38] (which later lead to the development of the `EISPACK` package). In subsequent

[1] Also available on line at `http://www.cs.ucdavis.edu/~bai/ET/contents.html`

codes based on density functional theory and planewave bases, diagonalization was replaced by "direct minimization", which in effect amounted to compute the subspace of minimum trace, i.e., an orthogonal basis $Q = [q_1, \ldots, q_{n_0}]$ such that $tr(Q^T AQ)$ is minimum. In fact, many publications of the mid 1990s focussed on avoiding orthogonality, which turned out to be hard to achieve. A method that was explicitlly based on "trace-minimization" was proposed by Sameh and Wisniewski [32] as far back as 1982. Many methods used in planewave codes, are variants of the same theme and are similar to subspace iteration and trace-min iteration. They begin with a certain subspace of size n_o (or close) and then improve each vector individually while the others are fixed. Clearly, when iterating on the i-th vector, orthogonality must be enforced against the first $i - 1$ vectors. While this does not refer directly to eigenvectors, the algorithm implicitly computes these eigenvectors individually.

Later, many codes offered an alternative to this type of scheme in the form of the Block-Davidson algorithm. When planewave bases are used, it is easy to precondition the eigenvalue problem for a number of reasons [25]. In real-space methods, the situation is different, and we found that preconditioning the eigenvalue problem is much harder [31]. The small gains one can expect from employing a preconditioner must be weighed against the loss of the 3-term recurrence of the Lanczos procedure. Specifically, one can potentially use the Lanczos procedure with an inexpensive form of reorthogonalization. This is no longer possible with the Davidson approach which requires a full orthogonalization at each step. In [4] we explored this approach. The Lanczos algorithm was adapted in a number of ways. First, we replaced the re-orthogonalization step by a partial reorthogonalization scheme [15, 24, 33, 34]. Then, individual eigenvectors were de-emphasized in the sense that they are not explicitly computed. Instead, a test based on the convergence of the desired trace was incorporated. This required only to compute the eigenvalues of the tridiagonal matrix at regular intervals during the iteration.

The method proposed in this paper exploits two distinct and complementary tools to solve the problem stated above. The first is a filtering technique which is used to dampen the undesirable contribution of the largest eigenvalues at each matrix-vector product in the Lanczos algorithm. This techniques employs a well-selected low pass filter polynomial, obtained via a conjugate residual-type algorithm in polynomial space. The second ingredient is the Lanczos algorithm with partial reorthogonalization. The main rationale for this approach is that filtering will help reduce the size of the Krylov subspace required for convergence, and this will result in substantial savings both in memory and in computational costs.

2 Krylov methods and the filtered Lanczos procedure

The Lanczos algorithm [14] (see also [6, 9, 24, 28]) is without a doubt one of the best known methods for computing eigenvalues and eigenvectors of very large and sparse matrices. The algorithm, which is deeply rooted in the theory of orthogonal polynomials, builds an orthonormal sequence of vectors which satisfy a 3-term recurrence. These vectors form an orthonormal basis $Q_m \in \mathbb{R}^{n \times m}$ of the Krylov subspace

$$\mathcal{K}_m(A, q_1) = \mathtt{span}\{q_1, Aq_1, A^2 q_1, \ldots, A^{m-1} q_1\}, \tag{3}$$

where q_1 is an arbitrary (typically random) initial vector with $\|q_1\| = 1$. An algorithmic outline of the method is shown in Fig. 1. Note that each step of the Lanczos algorithm requires the matrix A only in the form of matrix-vector products which can be quite appealing in some situations, such as when A is available in stencil form.

It is not difficult to verify that the sequence of vectors that are computed during the course of the Lanczos algorithm satisfy the 3-term recurrence

$$\beta_{i+1} q_{i+1} = Aq_i - \alpha_i q_i - \beta_i q_{i-1} \ . \tag{4}$$

The scalars β_{i+1} and α_i are computed to satisfy two requirements, namely that q_{i+1} be orthogonal to q_i and that $\|q_{i+1}\|_2 = 1$. As it turns out, this ensures, in theory at least, that q_{i+1} is orthogonal to $q_1, \ldots, q_i$. Therefore, in exact arithmetic the algorithm would build an orthonormal basis of the Krylov subspace, and this is very appealing. In practice, however, orthogonality is quickly lost after convergence starts taking place for one or a few eigenvectors [22].

Lanczos

*(*Input*)* Matrix $A \in \mathbb{R}^{n \times n}$, starting vector q_1, $\|q_1\|_2 = 1$, scalar m
*(*Output*)* Orthogonal basis $Q_m \in \mathbb{R}^{n \times m}$ of $\mathcal{K}_m(A, q_1)$,
unit norm vector q_{m+1} such that $Q_m^\top q_{m+1} = 0$

Set $\beta_1 = 0$, $q_0 = 0$
for $i = 1, \ldots, m$
  $w_i = Aq_i - \beta_i q_{i-1}$
  $\alpha_i = < w_i, q_i >$
  $w_i = w_i - \alpha_i q_i$
  $\beta_{i+1} = \|w_i\|_2$
  if $(\beta_{i+1} == 0)$ **then** `stop`
  $q_{i+1} = w_i / \beta_{i+1}$
end

Figure 1: The Lanczos algorithm. The inner product for vectors is denoted by $< .,. >$.

If $Q_m = [q_1, \cdots, q_m]$ and if T_m denotes the symmetric tridiagonal matrix

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{bmatrix}, \tag{5}$$

where the scalars α_i, β_i are computed by the Lanczos algorithm (Fig. 1), then the following matrix relation can be easily verified

$$AQ_m = Q_m T_m + \beta_{m+1} v_{m+1} e_m^\top . \tag{6}$$

Here e_m is the m-th column of the canonical basis and v_{m+1} is the last vector computed by the Lanczos algorithm.

The Lanczos algorithm approximates part of the spectrum of the original matrix A with that of the much smaller (typically $m \ll n$) and simpler tridiagonal matrix T_m. Of course, only a small number of the eigenvalues of A, typically only a small fraction of m, can be approximated by corresponding eigenvalues of T_m. This leads the discussion to the next section.

2.1 The Lanczos algorithm for computing large eigenspaces

As is well-known the Lanczos algorithm quickly yields good approximations to extremal eigenvalues of A. In contrast, convergence is typically much slower for the interior of the spectrum [24]. Figure 2 illustrates this phenomenon for a Hamitonian ($n = 17,077$) corresponding to the cluster $\mathtt{Si_{10}H_{16}}$ produced by a real space pseudopotential discretization code [5].

For the typical situation under consideration in this paper, an invariant subspace associated with a large number of the algebraically smallest eigenvalues of A is to be computed, reaching far into the interior of the spectrum. So we expect the number m of Lanczos steps required for all the desired eigenvectors to converge, to be very large. Therefore, if the Lanczos algorithm is to be applied without any form of restarting or preconditioning, then we will have to deal with two related constraints: (1) we will need to apply some form of reorthogonalization [4, 15, 24, 33, 34] to the Lanczos vectors and as a result (2) we need to store the Lanczos basis Q_m because it is needed by the reorthogonalization steps. The

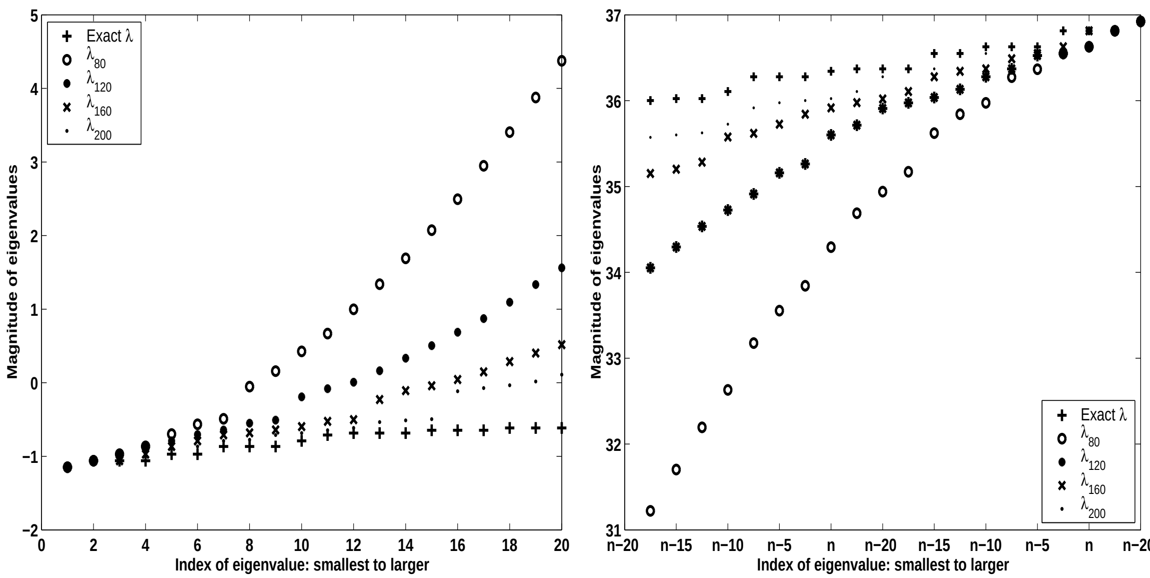

Figure 2: Convergence behavior for the 20 smallest (left) and the 20 largest (right) eigenvalues of the Hamiltonian ($n = 17077$) corresponding to $\texttt{Si}_{10}\texttt{H}_{16}$. Here, λ (denoted by $+$) are the exact eigenvalues of the Hamiltonian and $\lambda_m, m = 80, 120, 160$ and 200 are the corresponding eigenvalues obtained from the tridiagonal matrix T_m constructed by the Lanczos procedure.

first constraint increases computational cost and some care must be exercised for the reorthogonalization process not to become too expensive. The second, raises the issue of memory costs. Storing the Lanczos basis Q_m will require a large memory size, and may even force one to resort to secondary storage.

Note that reorthogonalization will ultimately require all basis vectors to be fetched in main memory and that the cost of orthogonalizing the vector q_k against all previous ones will incur a cost of $O(kn)$, which yields a quadratic total cost of $O(m^2n)$ when summed over m steps. This cost will eventually overwhelm any other computation done and it is the main reason why so many attempts have been made in the past to avoid or reduce the orthogonalization penalty in electronic structures codes, see, e.g., [7, 13, 19, 20].

Note also that there is an additional severe penalty due to memory traffic as the size of the system increases, because modern processors work at a much faster rate than memory subsystems. It was argued in [4] that memory requirements do not necessarily pose a significant problem for the matrix sizes encountered and the machines typically in use for large calculations. For example, storing 2,000 vectors of length 1 million, "only" requires 16 GB of memory, which is certainly within reach of most high-performance computers [2]. However, for larger calculations this could be a burden and out-of-core algorithms can help.

The above discussions strongly suggests that when computing large invariant subspaces, it is essential to use a Lanczos basis that is as small as possible. In order to achieve this, we apply the Lanczos process not on the original matrix A but rather on a matrix $\rho_l(A)$, where $\rho_l(t)$ is a polynomial of small degree designed to be close to zero for large eigenvalues and close to one for the eigenvalues of interest. Of course, polynomial acceleration in Krylov techniques is not a new idea (see for example [28] and references therein). Typically, the goal is to restart Lanczos after a fixed number of iterations with a starting vector from which unwanted eigendirections have been filtered out. In this paper we follow a slightly different approach. We do not employ any restarts, but rather filter each matrix-vector product in the Lanczos process using a small number of Conjugate-Residual type iterations on matrix A. As can be expected,

[2]In modern high-performance computers this will typically be available in a single node

the proposed scheme will require a much smaller number of basis vectors than without filtering. However, each matrix-vector product is now more costly. Experiments will show that the trade-off is in favor of filtering.

We motivate our approach by first describing a naive idea that does not work, and then sketch a remedy to make it work. The strategy discussed here is the basis for the proposed algorithm and it will be further refined throughout the paper. For the discussion we first consider the Lanczos algorithm in a hypotetical infinite precision implementation. Let $\lambda_1, \lambda_2, \cdots, \lambda_{n_o}, \cdots, \lambda_n$ be the eigenvalues of A and $u_1, u_2, \cdots, u_{n_o}, \cdots, u_n$ their associated eigenvectors. Suppose now that we are given an initial vector q_1 which is such that $< q_1, u_j >= 0$ for $j > n_o$, i.e., such that q_1 has no components along the directions $u_j, j > n_o$. Then it is known that the Krylov subspace $\mathcal{K}_{n_o}$ will be exactly the invariant subspace which we wish to compute, see, e.g., [28]. In other words, if the vector q_1 has zero components in the eigenvectors associated with the undesired eigenvalues $\lambda_{n_o+1}, \ldots, \lambda_n$, then in exact arithmetic,

$$\mathcal{K}_{n_o} = \mathcal{S}_{n_o}.$$

Of course, this is contingent on the condition that q_1 has nonzero components in the eigenvectors associated with the eigenvalues of interest $\lambda_1, \ldots, \lambda_{n_o}$, a property that will generally be true in practice. Then, the idea is the following: Find a polynomial $\rho_h(t)$ of high enough degree such that

$$\rho_h(\lambda_i) \approx \begin{cases} 1 & \text{for} \quad i \leq n_o \\ 0 & \text{for} \quad i > n_o \end{cases}$$

and then filter the vector q_1 with this polynomial before applying the Lanczos procedure. In other words, redefine $q_1 := \rho_h(A)q_1$ and apply n_o steps of the Lanczos algorithm with this filtered q_1 as initial guess to obtain $\mathcal{K}_{n_o} = \mathcal{S}_{n_o}$. The polynomial ρ_h is a low-pass filter-polynomial and an actual example is shown in Figure 3.

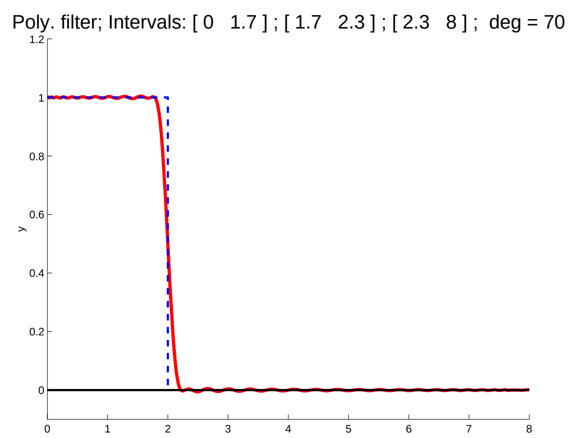

Figure 3: A sample filter polynomial of degree 70 which is built to be close to 1 in [0, 1.7], and 0 in [2.3, 8].

One can imagine performing a practical Lanczos algorithm with this approach, using partial or full reorthogonalization. This, however, does not work because when the initial vector q_1 has no weight in the largest eigenvectors, the Lanczos polynomial will tend to be extremely large in the interval of the undesired eigenvalues and this makes the whole procedure rather unstable.

In this paper we propose a straightforward remedy which, as it will be shown, works very well in practice. It consists of replacing matrix A in the Lanczos procedure by the matrix $B = \rho_l(A)$, where ρ_l is designed to have the same desirable characteristics as ρ_h but is of much smaller degree. Observe that

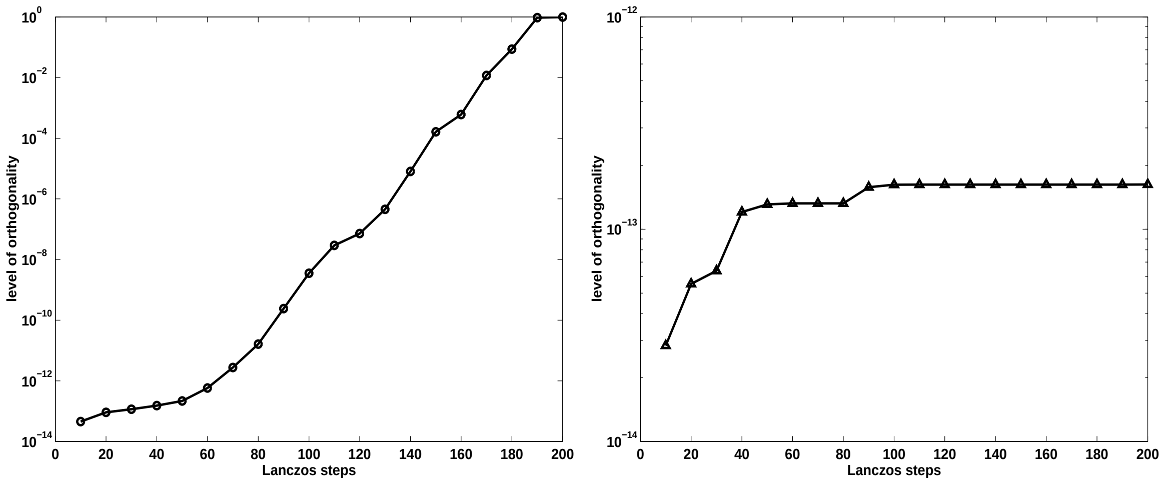

Figure 4: Levels of orthogonality of the Lanczos basis for the Hamiltonian ($n = 17077$) corresponding to $\texttt{Si}_{10}\texttt{H}_{16}$. Left: Lanczos without reorthogonalization. Right: Lanczos with partial reorthogonalization. The number of reorthogonalizations was 34 with an additional 3400 inner vector products.

since $\rho_l(t)$ is small for $t > \lambda_{n_o}$, $\rho_l(A)$ will not amplify any components associated with $\lambda_j, j > n_o$. In fact, $\rho_l(A)$ will act as if there are no components at all in the undesired subspace. This is, in a nutshell, the heart of the proposed algorithm. Essentially, our goal is to trade a long Lanczos basis and the associated expensive reorthogonalization steps with additional matrix-vector products induced by the application of $\rho_l(A)$ at every step of Lanczos.

Given the high cost of orthogonalizing large bases, it is not difficult to imagine why this approach can be economical. If A has μn nonzero entries and if ρ_l is of small degree, say 8, then each matrix-vector product with $B = \rho_l(A)$ will cost about $O(8\mu n)$ floating point operations. In contrast, an orthogonalization step costs $O(2in)$ operations at step i. If i is 1,000, and $\mu = 10$, we have a cost of $2,000n$ versus about $80n$ for the new approach. Though this may be compelling enough in the case of full reorthogonalization, the situation is more complex to analyze when a form of partial reorthogonalization is used. However, experiments (see Section 5) show that the procedure is superior to the straightforward use of the Lanczos procedure with partial reorthogonalization: it saves not only memory but arithmetic as well.

2.2 The partially reorthogonalized Lanczos procedure

As is well-known, the standard Lanczos process is “unstable” in the sense that the column vectors of Q_m will suddenly start loosing their orthogonality [22]. As an illustration, consider again the Hamiltonian ($n = 17,077$) corresponding to $\texttt{Si}_{10}\texttt{H}_{16}$. We test the orthogonality of the bases $Q_i, i = 1, \ldots, m$, with $m = 200$ by computing the norm $\|Q_i^\top Q_i - I_i\|_2$, where I_i is the identity matrix of size i. The left plot in Figure 4 illustrates the rapid deterioration of orthogonality among basis vectors.

A number of existing reorthogonalization schemes are often employed to remedy the problem. The simplest of these consists of a full reorthogonalization approach, whereby the orthogonality of the basis vector q_i is enforced against all previous vectors at each step i. This means that the vector q_i, which in theory is already orthogonal against $q_1, \ldots, q_{i-1}$, is orthogonalized (a second time) against these vectors. In principle we no longer have a 3-term recurrence but this is not an issue as the corrections are small and usually ignored (see however Stewart [37]). The additional cost at the m-th step will be $O(nm)$. So, if reorthogonalization is required at each step, then we require an additional cost of $O(nm^2)$ which is consumed by the Gram-Schmidt process. In general, all basis vectors need to be available in main memory, making this approach impractical for bases of large dimension.

An alternative to full reorthogonalization is *partial reorthogonalization* which attempts to perform reorthogonalization steps only when they are deemed necessary. The goal is not so much to guarantee that the vectors are exactly orthogonal, but to ensure that they are at least nearly orthogonal. Typically, the loss of orthogonality is allowed to grow to roughly the square root of the machine precision, before a reorthogonalization is performed. A result by Simon ([33]) ensures that we can get fully accurate approximations to the Ritz values (eigenvalues of the tridiagonal matrix T_m) in spite of a reduced level of orthogonality among the Lanczos basis vectors. Furthermore, a key to the successful utilization of this result is the existence of clever recurrences which allow us to estimate the level of orthogonality among the basis vectors [15, 34]. It must be stressed that the cost of updating the recurrence is very modest. Let $\omega_{i,j} = q_i^\top q_j$ denote the "loss of orthogonality" between any basis vectors q_i and q_j. Then, the following is the so called ω-recurrence [34]:

$$\beta_i \omega_{i+1,j} = (\alpha_j - \alpha_i)\omega_{i,j} + \beta_{j-1}\omega_{i,j-1} - \beta_{i-1}\omega_{i-1,j}, \tag{7}$$

where the scalars α_i and $\beta_i, i = 1, \ldots,$ are identical to the ones computed by the Lanczos algorithm (see Fig. 1).

Thus, we can cheaply and efficiently probe the level of orthogonality of the current vector (say q_i) and determine whether a reorthogonalization step against previous basis vectors is required. The left plot in Figure 4 illustrates the corresponding level of orthogonality when partial reorthogonalization is applied. Observe that only 34 reorthogonalization steps are required, compared to 200 that would have been required if full reorthogonalization was employed.

It was shown in [4] that partially reorthogonalized Lanczos combined with techniques that avoid explicit computation of eigenvectors can lead to significant savings in computing charge densities for electronic structure calculations. Partial reorthogonalization will play a key role in the algorithm to be described in the next section.

2.3 Polynomial acceleration and restarting techniques

Classical convergence theory for Lanczos (see [11], [27]) predicts that if $\theta(\psi_i, q_1)$ is the acute angle between the starting vector q_1 and the eigenvector ψ_i associated with i-th eigenvalue λ_i, then

$$\tan\theta(\psi_i, \mathcal{K}_m) \le \frac{K_i}{T_{m-i}(\gamma_i)} \tan\theta(\psi_i, q_1), \tag{8}$$

where

$$\theta(\psi_i, \mathcal{K}_m) = \arcsin \frac{\|(I - Q_m Q_m^\top)\psi_i\|}{\|\psi_i\|}$$

is the acute angle between the eigenvector ψ_i and the Krylov subspace $\mathcal{K}_m$ and

$$\gamma_i = 1 + 2\frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n} \tag{9}$$

$$K_i = \prod_{i=1}^{i-1} \frac{\lambda_j - \lambda_n}{\lambda_j - \lambda_i}, \quad \text{if} \ \ i \ne 1 \ \ \text{and} \ \ K_1 = 1. \tag{10}$$

With $T_k(t)$ we denote the Chebyshev polynomial of the first of kind of order k, $T_k(t) = \cos(k \arccos(t))$. This result suggests that in order for the Krylov subspace $\mathcal{K}_m$ to converge to the wanted invariant subspace, the initial vector q_1 must be "rich" in eigendirections associated with the wanted eigenvalues $\lambda_1, \ldots, \lambda_{n_o}$. The rate of convergence depends on the "gap" $|\lambda_{n_o} - \lambda_{n_o+1}|$ and on the "spread" of the spectrum $|\lambda_1 - \lambda_n|$.

Observe that in exact arithmetic, if the starting vector q_1 is orthogonal to an eigenvector ψ_j, then the Krylov subspace $\mathcal{K}_m$ will never have any components in ψ_j, regardless of the number of steps m. Restarting techniques utilize this property to speed up the computation of the desired invariant subspace. The goal is to progressively construct a starting vector $q_1^{(k)}$, which at each restart k will have larger components in desired eigendirections, and smaller ones in undesired eigendirections. In contrast to the standard Lanczos procedure, the dimension of the Kyrlov subspace is not allowed to grow indefinitely. When a maximum number of iterations m is reached, a new starting vector $q_1^{(k+1)}$ is selected and the process is restarted.

Restarting can be either explicit or implicit. In explicit restarting (see for example [28]) the next starting vector $q_1^{(k+1)}$ is selected by applying a certain polynomial $\rho(t)$ on A such that $q_1^{(k+1)} = \rho(A)q_1^{(k)}$. The polynomial $\rho(t)$ can be selected in order to minimize a discrete norm on the set of eigenvalues of the tridiagonal matrix $T_m^{(k)}$, or in order to minimize a continuous norm in a certain interval. In both cases the goal is to filter out of the next starting vector any undesired eigendirections.

In implicit restarting the goal of properly filtering the next starting vector remains the same, however there is not an explicit application of the polynomial filter (see [36]). Instead, a sequence of implicit QR steps is applied to the tridiagonal matrix $T_m^{(k)}$ resulting to a new Lanczos factorization of length $p < m$

$$AQ_p^+ = Q_m^+ T_p^+ + F^+, \tag{11}$$

which is the one that would have been obtained with starting vector $q_1^{(+)} = \rho(A)q_1^{(k)}$. The roots of polynomial $\rho(t)$ can be selected to be eigenvalues of $T_m^{(k)}$ (exact shifts). However, other alternatives have been considered, such as Leja points, refined shifts, and harmonic Ritz values [1, 10, 23, 35] to name a few. Factorization (11) is expanded to a maximum length m by applying $m-p$ additional Lanczos steps, resulting to the factorization

$$AQ_m^{(k+1)} = Q_m^{(k+1)} T_m^{(k+1)} + \beta_{m+1}^{(k+1)} v_{m+1} e_m^\top. \tag{12}$$

Thus, restarting can be designed to filter out eigendirections corresponding to eigenvalues $\lambda_j > \lambda_{n_o}$. The goal is to accelerate convergence towards the algebraically smallest eigenvalues. However, round-off will cause eigendirections in the largest eigenvalues to quickly reappear into the iteration. This is illustrated in Fig. 5. The matrix that is tested corresponds to a second order finite differences approximation of the two dimensional Laplace differential operator. The starting vector is the sum

$$q_1 = \sum_{k=1}^{n_o} \psi_i$$

of the eigenvectors corresponding to the smallest $n_o = 200$ eigenvalues of the matrix. The left plot of Figure 5 illustrates that at the first step of Lanczos vector q_1 is orthogonal (up to machine precision) to the unwanted eigenvectors. However, it only takes $m = 13$ steps of Lanczos for the coefficients in the largest eigenvectors to start dominating the last basis vector q_m.

What happened can be easily explained. Let ϵ denote the machine precision. Assume that $< q_1, phi_i >= \epsilon$, for a given eigenvector phi_i with $i > n_o$. Recall that the Lanczos vector q_{m+1} is of the form $q_{m+1} = p_m(A)q_1$ where p_m is a polynomial of degree m, called the $(m+1)-st$ Lanczos polynomial. The sequence of polynomials p_k is orthogonal with respect to a certain discrete inner product. Since the initial vector has very small components in the eigenvectors associated with eigenvalues $i > n_o$, it is to be expected that the Lanczos polynomial, p_m is such that $p_m(\lambda_i) \gg 1$ for $i > n_o$. Therefore, we will have

$$\begin{aligned} < q_{m+1}, \psi_i > &= < p_m(A)q_1, \psi_i > \\ &= < q_1, p_m(A)\psi_i > \\ &= p_m(\lambda_i) < q_1, \psi_i > \\ &= p_m(\lambda_i)\epsilon. \end{aligned} \tag{13}$$

As a result the small component ϵ will be amplified by the factor $p(\lambda_i)$ which is likely to be very large.

The situation can be remedied by replacing A by an operator of the form $B = \rho(A)$ where $\rho(\lambda_i)$ is small. If B is used in the Lanczos algorithm, then note that every time we multiply q by B, a component in the direction ψ_i that is small (relative to the others), will remain small.

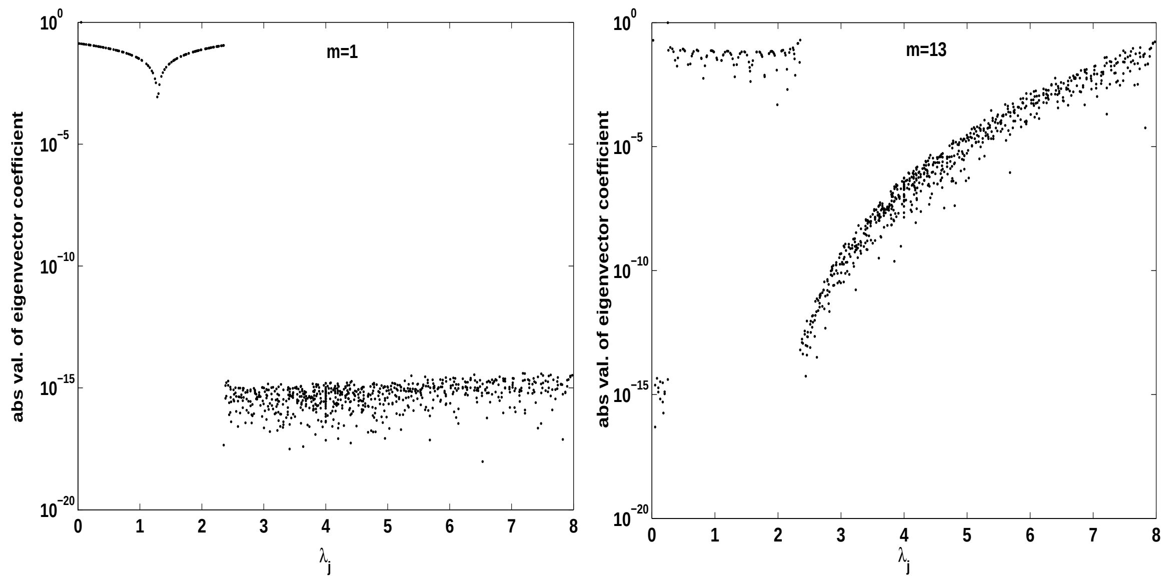

Figure 5: Coefficients of the last basis vector q_m of the Lanczos procedure (no partial reorthogonalization was required) for the discretization of the Laplacean, when the starting vector does not have any components in undesired eigenvectors. Left: one step. Right: $m = 13$ steps.

Before we state the result with detail, we must recall that in inexact arithmetic, the Lanczos relation (4) is replaced by a relation of the form:

$$Aq_i = \beta_{i+1} q_{i+1} + \alpha_i q_i + \beta_i q_{i-1} - z_i \ . \tag{14}$$

where z_i is an error vector which, in general, remains small.

Lemma 2.1 *Consider any eigenvalue $\lambda > \lambda_{n_o}$ and let ψ be its associated eigenvector and $\delta \equiv \rho(\lambda)$. Assume that the sequence $\{q_i\}$ satisfies the model (14) and define $\epsilon_i^{\psi} =< \psi, z_i >$. Then the scalar sequence $\sigma_i =< q_i, \psi >$ satisfies the recurrence,*

$$\beta_{i+1}\sigma_{i+1} + (\alpha_i - \delta)\sigma_i \ + \ \beta_i \sigma_i = \epsilon_i^{\psi} \tag{15}$$

and, assuming $\beta_{m+1} e_1^{\top} (T_m - \delta I)^{-1} e_m \neq 0$ then the component σ_{m+1} of q_{m+1} along ψ, can be expressed as

$$\sigma_{m+1} = \frac{\varepsilon_m^{\top} (T_m - \delta I)^{-1} e_1 - \sigma_1}{\beta_{m+1} e_m^{\top} (T_m - \delta I)^{-1} e_1}, \tag{16}$$

in which $\varepsilon_m = [\epsilon_1^{\psi}, \epsilon_2^{\psi}, \cdots, \epsilon_m^{\psi}]^{\top}$ and T_m is the tridiagonal matrix (5).

Proof. We begin with the relation

$$Bq_i = \beta_{i+1} q_{i+1} + \alpha_i q_i + \beta_i q_{i-1} - z_i \ .$$

Taking the inner product with ψ yields

$$< Bq_i, \psi >= \beta_{i+1} < q_{i+1}, \psi > + \alpha_i < q_i, \psi > + \beta_i < q_{i-1}, \psi > - \epsilon_i^{\psi}.$$

Since $B\psi = \delta\psi$, this readily yields the expression (15).

Define the vector $s_m = [\sigma_1, \sigma_2, \cdots, \sigma_m]^\top$. We can rewrite the relations (15) for $i = 1, \ldots, m$ in matrix form as

$$(T_m - \delta I)s_m = \varepsilon_m - \beta_{m+1}\sigma_{m+1}e_m$$

which yields the relation, $s_m = (T_m - \delta I)^{-1}\varepsilon_m - \beta_{m+1}\sigma_{m+1}(T_m - \delta I)^{-1}e_m$. Now, we add the condition that σ_1 is known:

$$\sigma_1 = e_1^\top s_m = e_1^\top (T_m - \delta I)^{-1}\varepsilon_m - \beta_{m+1}\sigma_{m+1}e_1^\top (T_m - \delta I)^{-1}e_m,$$

from which we obtain the desired expression (16). ■

The main point of the above lemma is that it explicitly provides the amplification factor for the coefficient in the direction ψ, in terms of computed quantities. This factor is the denominator of the expression (16). Note that in exact arithmetic, the vector ε_m is zero and the initial error of σ_1 in the direction of ψ is divided by the factor $\beta_{m+1}e_1^\top (T_m - \delta I)^{-1}e_m$. We can obtain a slightly simpler expression by "folding" the term σ_1 into the vector ε_m. This is helpful if σ_1 is of the same order as the ϵ_i^ψs as it simplifies the expression. Set

$$\hat{\varepsilon}_m = \varepsilon_m - \sigma_1(T_m - \delta I)e_1 \ .$$

Note that only ϵ_1^ψ and ϵ_2^ψ are modified into $\hat{\epsilon}_1^\psi = \epsilon_1^\psi - (\alpha_1 - \delta)\sigma_1$ and $\hat{\epsilon}_2^\psi = \epsilon_2^\psi - \beta_2\sigma_1$, while the other terms remain unchanged, i.e., $\hat{\epsilon}_i^\psi = \epsilon_i^\psi$ for $i > 2$. Then, (16) becomes,

$$\sigma_{m+1} = \frac{\hat{\varepsilon}_m^\top (T_m - \delta I)^{-1}e_1}{\beta_{m+1}e_m^\top (T_m - \delta I)^{-1}e_1} \ . \qquad (17)$$

Let us consider the unfavorable scenario first. When $B \equiv A$ then T_m is simply the tridiagonal matrix obtained from the Lanczos algorithm and δ is an eigenvalue of A. Assume that $\lambda = \lambda_n$, the largest (unwanted) eigenvalue. Even if q_1 has very small components in the direction of λ, convergence will eventually take place, see (13), and T_m will tend to have an eigenvalue close to λ, so $(T_m - \delta I)^{-1}e_1 \equiv y_m$ is close to an eigenvector of T_m associated with its largest eigenvalue. As is well known, the last components of (converged) eigenvectors of T_m will tend to be much smaller than the first ones. Therefore, if $\hat{\varepsilon}_m$ is a small random vector, then σ_m will become larger and larger because the numerator will converge to a certain quantity while the denominator will converge to zero.

The use of an inner polynomial $\rho(t)$ prevents this from happening early by *ensuring that convergence towards unwanted eigenvalues does not take place*. In this situation δ is an eigenvalue of B among many others that are clustered around zero, so convergence is considerably slower toward the corresponding eigenvector. By the time convergence takes place, the desirable subspace will have already been computed.

3 The filtered Lanczos procedure

Partial reorthogonalization can significantly extend the applicability of Lanczos in electronic structure calculations (see [4]), but there are computational issues related to the use of very long Lanczos bases when a large invariant subspace is sought. These issues can be addressed by employing polynomial filtering in the Lanczos procedure.

In exact arithmetic, the ideal solution to this problem is to use an initial vector which is filtered so that it has no eigencomponents associated with $\lambda_i, i > n_o$. However, we saw earlier that in the course of the Lanczos procedure, components along the largest eigenvectors will quickly return. We discussed the reasons for this behavior and suggested a simple remedy which consists of replacing the matrix-vector product Aq_i of the Lanczos algorithm (see line 3, Fig. 1) by $\rho_l(A)q_i$, where $\rho_l(t)$ is a low degree polynomial filter that approximates the Heaviside function (see Fig. 6). The interval $[\gamma,\ \beta]$ contains all the unwanted (largest) eigenvalues, which are approximately mapped by $\rho_l(t)$ to zero.

All that is required to implement the proposed filtered Lanczos scheme is to substitute the matrix-vector product Aq_i with a function $\mathcal{P}(A, q_i, d)$ which evaluates the product of the matrix polynomial

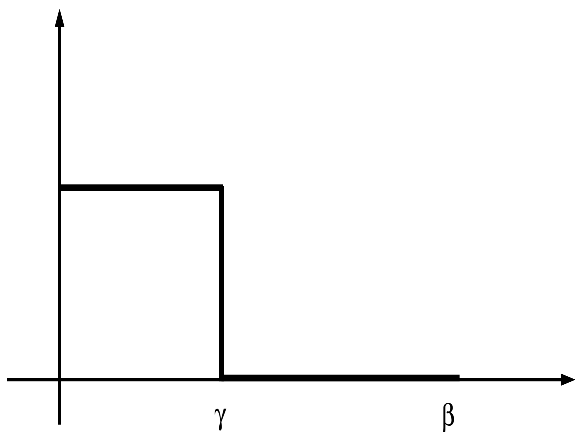

Figure 6: The Heaviside function for the interval $[\gamma,\ \beta]$.

$\rho_l(A)$ with the vector q_i. Let d be the degree of the polynomial $\rho_l(t)$. Then, the cost per step of the filtered Lanczos procedure, compared with the plain Lanczos procedure, is d additional matrix-vector products.

Observe that the filtered Lanczos process constructs an approximate invariant subspace for matrix $\rho_l(A)$ which is also an invariant subspace for A itself. However, although the restriction of $\rho_l(A)$ on the orthogonal Lanczos basis Q_m will be a tridiagonal matrix, this is no longer true for A. Thus, while

$$Q_m^\top \rho_1(A) Q_m = T_m, \tag{18}$$

where T_m is tridiagonal and defined as in (5), for A we have that

$$Q_m^\top A Q_m = \tilde{T}_m, \tag{19}$$

where $\tilde{T}_m$ is in general dense. The eigenvalues of A are approximated by those of $\tilde{T}_m$, while the eigenvalues of T_m approximate those of $\rho_l(A)$. However, A and $\rho_l(A)$ have the same eigenvectors. Thus, if we consider the matrix of normalized eigenvectors Y of T_m and $\tilde{Y}$ of $\tilde{T}_m$ respectively, then approximations to the eigenvectors of A are given either by the columns of the matrices $Q_m Y$ or $Q_m \tilde{Y}$. Furthermore, approximations to the eigenvalues of A are available from the eigenvalues of

$$\hat{T}_m = Y^\top Q_m^\top A Q_m Y. \tag{20}$$

Similarly to the Lanczos procedure, the basis vectors q_i in the filtered Lanczos procedure are also expected to rapidly loose orthogonality. Thus, the partial reorthogonalization techniques of section 2.2 will prove to be particularly useful in the practical deployment of the method.

The larger the degree of the polynomial $\rho_l(t)$, the closer it can be made, to the Heaviside function. On the other hand, using a larger degree d will induce a higher computational cost. It is important to note that in practice we do not seek to approximate the Heaviside function everywhere on its domain of definition. We would like the polynomial $\rho_l(t)$ to take small values on the region of the unwanted eigenvalues. The goal is for the corresponding eigendirections, that have been with high accuracy removed from the starting vector q_1, not to be significantly amplified during the course of Lanczos. Section 4 discusses a Conjugate Residual type iteration that achieves the aforementioned goal. For the time being, it suffices to consider the application of the filtering polynomial as a "black box" function $\mathcal{P}(A, q_i, d)$, to describe the Filtered Lanczos iteration.

3.1 The algorithm

In order to compute a basis for an invariant subspace $\mathcal{S}_{n_o}$ for the n_o algebraically smallest eigenvalues of matrix A, we assume that we are given an interval $(\gamma,\ \beta]$, which contains all the unwanted eigenvalues $\lambda_j > \lambda_{n_o}$. Assuming that matrix A does not have any negative eigenvalues, it suffices to consider only the left endpoint γ of the interval. In electronic structure calculations, the problem is often a variation of this one, in that we wish to compute an invariant subspace associated with the n_o smallest eigenvalues. However, there is an outer loop and previous information can be used to obtain a good interval on which to restrict the search. There are also instances where the number of eigenvalues n_o is unknown, but rather we are given an upper bound γ for the eigenvalues that need to be considered.

Starting vector. It is important that the starting vector q_1 be free of components in the undesired eigenvectors. To this end we apply a high degree polynomial filter ρ_h on a random vector $\tilde{q}$, such that $q_1 = \rho_h(A)\tilde{q}$. The degree of this first polynomial can be quite high (say $d_h = 200$ or so) to get a good elimination of the undesired components.

Bounding interval. If we are not given an interval $[\alpha,\ \beta]$ that tightly contains the eigenvalues, then we employ a number of unrestarted Lanczos iterations in order to obtain approximations for the bounds α and β. In practice, the number of these iterations is kept low. Let r_1 and r_n be the residual vectors for the approximate extremal eigenvalues $\tilde{\lambda}_1$ and $\tilde{\lambda}_n$ of matrix A after a number of Lanczos iterations. Then, we use the practical bounds $\tilde{\alpha} = \tilde{\lambda}_1 - \|r_1\|$ and $\tilde{\beta} = \tilde{\lambda}_n + \|r_n\|$. If, $\tilde{\alpha}$ (or α) is negative, then we shift the Hamiltonian so as to ensure that all its eigenvalues are positive.

Inner polynomial transformation. The main Lanczos iteration will be performed with a filter polynomial of A, i.e., the Lanczos algorithm is run with $B = \rho_l(A)$. The degree d_l of ρ_l is much smaller than that of ρ_h, in order to reduce the overall cost. Typically $d_l \equiv 8$.

Convergence criterion. Equally important in order to restrain the computational cost is the convergence test. Let $(\tilde{\lambda}_i, \tilde{x}_i)$ be an approximate eigenpair, where

$$x_i = Q_m y_i$$

and $(\tilde{\lambda}_i, y)$ is an eigenpair of the dense matrix (19). Then, it is natural to monitor the norm of the remainder $r_i = A\tilde{x}_i - \tilde{\lambda}_i \tilde{x}_i$. It is well-known (see, e.g., [24]) that

$$\|r_i\| = \|A\tilde{x}_i - \tilde{\lambda}_i \tilde{x}_i\| = |\beta_{m+1}|\ |y_i^m|,$$

where y_i^m is the last element of the eigenvector y_i. Monitoring the norm $\|r_i\|$ for all n_o eigenvalues of interest, at each step of the (filtered) Lanczos procedure entails computing the eigenvectors y_i every time a test is required. This, however, is expensive, since computing all eigenpairs of a dense $k \times k$ matrix will be cubic in k, and this is to be summed for a total of m Lanczos steps.

We have chosen a significantly cheaper alternative. We monitor the sum of the eigenvalues of matrix $\tilde{T}_i$, that are smaller than the upper bound γ, $s_i = \sum_{\tilde{\lambda}_i < \gamma} \tilde{\lambda}_i$. When s_i converges we stop the iteration. In order to further restrict the cost we conduct the convergence test at fixed intervals and not at every step of the Filtered Lanczos iteration. If convergence is achieved after m steps, then the cost of the test will be $O(m^2)$.

Computation of the projection matrix $\tilde{T}_m$. Observe that

$$\tilde{T}_i = Q_{i+1}^\top A\ Q_{i+1} = [\ Q_i \quad q_{i+1}\ ]^\top A [\ Q_i \quad q_{i+1}\ ] = \begin{bmatrix} Q_i^\top A Q_i & Q_i^\top A q_{i+1} \\ q_{i+1}^\top A Q_i & q_{i+1}^\top A q_{i+1} \end{bmatrix}. \tag{21}$$

Thus, matrix $\tilde{T}_m$ can be computed incrementally during the course of the algorithm. Obviously, if $\tilde{T}_m$ is updated at every step i, then no additional memory is required. However, a more efficient `BLAS 3`

Filtered Lanczos

*(*Input*)*
Matrix $A \in \mathbb{R}^{n\times n}$, starting vector q_1, $\|q_1\|_2 = 1$, scalar m,
polynomial filter function $\mathcal{F}(A, q, d)$ that approximates the step function,
low and high polynomial degrees d_l, d_h, stride `strd`, upper bound γ

*(*Output*)*
Eigenvalues of A smaller than γ and orthogonal basis $Q = [q_1, q_2, \ldots]$ for the invariant subspace associated with these eigenvalues

Set $\beta_1 = 0$, $q_0 = 0$
Thoroughly filter initial vector $q_1 = \mathcal{F}(A, q_1, d_h)$, $q_1 = q_1/\|q_1\|$
for $i = 1, \ldots, m$
  $w_i = \mathcal{F}(A, q_i, d_l) - \beta_i q_{i-1}$
  $\alpha_i = < w_i, q_i >$
  $w_i = w_i - \alpha_i q_i$
  $\beta_{i+1} = \|w_i\|_2$
  if $(\beta_{i+1} == 0)$ **then** `stop`
  $q_{i+1} = w_i/\beta_{i+1}$
  if `rem`$(i, strd) == 0$ **then**
    Compute last row/column of matrix $\tilde{T}_i = Q_i^\top A Q_i$
    Compute all eigenvalues $\tilde{\lambda}_j$ of $\tilde{T}_i$ such that $\tilde{\lambda}_j < \gamma$
    Compute $s_i = \sum_{\tilde{\lambda}_i < \gamma} \tilde{\lambda}_i$
    if $(|(s_i - s_{i-1})/s_{i-1}| <$ `tol`$)$ **then** `break`
  end
end

Figure 7: The Filtered Lanczos algorithm. The inner product for vectors is denoted by $< .,. >$.

implementation is possible if we postpone the update of $\tilde{T}_m$ and rather perform it at fixed intervals (which can be made to coincide with the intervals at which convergence it checked). This will come at the expense of a few additional vectors in memory. In particular, we will have to store the vectors Aq_{i+1} for a number of consecutive steps.

Figure 7 shows a high level algorithmic description of the Filtered Lanczos iteration.

4 Polynomial filters

This section focuses on the problem of defining the polynomial filter and applying it. Details on the algorithms described here can be found in [30]. We begin with a brief summary of filtering techniques when solving linear systems of equation by "regularization" [21]. In regularized solution methods, one seeks to find an approximate solution to the linear system $Ax = b$ by inverting A only in the space associated with the largest eigenvalues, leaving the other part untouched. As was explained in [30], computing a filtered solution amounts to computing a vector $s(A)b$ whose residual vector $\rho(A)b = b - As(A)b$ is a certain filter polynomial, typically one that is computed to be close to one for small eigenvalues and close to zero for larger eigenvalues. In other words, it would resemble the desired filter polynomial, such as the one shown in Figure 3.

The approximate solutions produced by Krylov subspace methods for solving a linear system $Ax = b$, are of the form $s_j(A)r_0$ where s_j is a polynomial of degree $\leq j$. The corresponding residual vector

Generic Conjugate Residual Algorithm

Compute $r_0 := b - Ax_0$, $p_0 := r_0$,	$\pi_0 = \rho_0 = 1$
	Compute $\lambda\pi_0$
for $j = 0, 1, \ldots,$ until convergence :	
$\alpha_j := < \rho_j, \lambda\rho_j >_g \, / < \lambda\pi_j, \lambda\pi_j >_g$	
$x_{j+1} := x_j + \alpha_j p_j$	
$r_{j+1} := r_j - \alpha_j A p_j$	$\rho_{j+1} = \rho_j - \alpha_j \lambda \pi_j$
$\beta_j := < \rho_{j+1}, \lambda\rho_{j+1} >_g < \rho_j, \lambda\rho_j >_g$	
$p_{j+1} := r_{j+1} + \beta_j p_j$	$\pi_{j+1} := \rho_{j+1} + \beta_j \pi_j$
	Compute $\lambda\pi_{j+1}$
end	

Figure 8: Generic Conjugate Residual algorithm

is $\rho_{j+1}(\lambda) = 1 - \lambda s_j(\lambda)$. This polynomial is of degree $j+1$ and has value one at $\lambda = 0$. In standard (unfiltered) methods one attempts to make the polynomial $\lambda s_j(\lambda)$ close to the function 1 on the (discrete) set of eigenvalues. Chebyshev methods attempt to make the polynomial $\lambda s(\lambda)$ close to the function 1, uniformly, on the (continuous) set $[\alpha,\ \beta]$ containing the spectrum (with $0 < \alpha < \beta$). A number of other methods have been developed which attempt to make the polynomial $\lambda s(\lambda)$ close to the function 1, in some least-squares sense, on the interval $[\alpha,\ \beta]$.

In the standard Conjugate Residual algorithm (see, e.g., [29]), the solution polynomial s_j minimizes the norm $\|(I - As(A))r_0\|_2$ which is nothing but a discrete least-squares norm when expressed in the eigenbasis of A:

$$\|(I - As(A))r_0\|_2 = \left[\sum_1^N (1 - \lambda_i s(\lambda_i))^2\right]^{1/2} \equiv \|1 - \lambda s(\lambda)\|_D .$$

It is possible to write a CR-like algorithm which minimizes $\|1 - \lambda s(\lambda)\|_g$ for any least-squares norm associated with a (proper) inner product of polynomials

$$< p, q >_g \ .$$

The related generic CR algorithm is given in Figure 8

It can be easily shown that the residual polynomial ρ_j generated by this algorithm minimizes $\|\rho(\lambda)\|_g$ among all polynomials of the form $\rho(\lambda) = 1 - \lambda s(\lambda)$, where s is any polynomial of degree $\leq j-1$. In other words, ρ_j minimizes $\|\rho(\lambda)\|_g$ among all polynomials ρ of degree $\leq j$, such that $\rho(0) = 1$. In addition, the polynomials $\lambda\pi_j$ are orthogonal to each other.

In order to add filtering to the above algorithm, note that filtering amounts to minimizing some norm of $\phi(\lambda) - \lambda s(\lambda)$, where ϕ is the given filter function. One must remember that $\phi(A)v$ is not necessarily easy to evaluate for a given vector v. In particular, $\phi(A)r_0$ may not be available.

The relation between regularized filtered iterations and polynomial iterations, such as the one we are seeking for the eigenvalue problem, may not be immediately clear. Observe that the residual polynomial $\rho_m(t)$ can be used as a filter polynomial for a given iteration. For example, the residual polynomial shown in Figure 3, which is of the form $\rho(\lambda) = 1 - \lambda s(\lambda)$, can be used for computing all eigenvalues in the interval $[0,\ 1.7]$. The dual filter $1 - \rho(\lambda)$ has small values in $[0,\ 1.7]$ and it can be used to compute the invariant subspace associated with the eigenvalues in the interval $[2.3,\ 8]$, though this may possibly require a large subspace. Notice that one of the main difficulties with this class of techniques is precisely the issue of the dimension of the subspace, as there is no inexpensive way of knowing in advance how many eigenvalues there are in a given interval.

4.1 Corrected CR algorithm

The standard way of computing the best polynomial is to generate an orthogonal sequence of polynomials and expand the least-squares solution in it. This approach was taken in [8] and more recently in [12].

The formulation of the solution given next is based on the following observation. The polynomials associated with the residual vectors of the (standard) CR algorithm, are such that $\{\lambda\pi_j\}$ is an orthogonal sequence of polynomials and so it can be used as an intermediate sequence in which to express the solution. We can generate the residual polynomial which will help obtain the p_i's: *the one that would be obtained from the actual CR algorithm*, i.e., the same r vectors as those of the generic CR algorithm (see Fig. 8). It is interesting to note that with this sequence of residual vectors, which will be denoted by $\tilde{r}_j$, it is easy to generate the directions p_i *which are the same* for both algorithms. The idea becomes straightforward: obtain the auxiliary residual polynomials $\tilde{\rho}_j$ that are those associated with the *standard* CR algorithm and exploit them to obtain the π_i's in the same way as in the CR algorithm. The polynomials $\lambda\pi_j$ are orthogonal and therefore the expression of the desired approximation is the same. The algorithm is described in Figure 9 where now $\tilde{\rho}_j$ is the polynomial associated with the auxiliary sequence $\tilde{r}_j$.

The only difference with a generic Conjugate Residual-type algorithm (see, e.g. Figure 8) is that the updates to x_{j+1} use different coefficients α_j from the updates to the vectors $\tilde{r}_{j+1}$. Observe that the residual vectors $\tilde{r}_j$ obtained by the algorithm are just auxiliary vectors that do not correspond to the actual residuals $r_j = b - Ax_j$. Needless to say, these actual residuals, the r_j's, can also be generated after Line 5 (or 6) from $r_{j+1} = r_j - \alpha_j Ap_j$. Depending on the application, it may or may not be necessary to include these computations.

The solution vector x_{j+1} computed at the j-th step of the corrected CR algorithm is of the form $x_{j+1} = x_0 + s_j(A)r_0$, where s_j is the j-th degree polynomial:

$$s_j(\lambda) = \alpha_0\pi_0(\lambda) + \cdots + \alpha_j\pi_j(\lambda) \ . \tag{22}$$

The polynomials π_j and the auxiliary polynomials $\tilde{\rho}_{j+1}(\lambda)$ satisfy the orthogonality relations,

$$\langle \lambda\pi_j(\lambda), \lambda\pi_i(\lambda)\rangle_w = \langle \lambda\tilde{\rho}_j(\lambda), \tilde{\rho}_i(\lambda)\rangle_w = 0 \quad \text{for} \quad i \neq j \ . \tag{23}$$

In addition, the filtered residual polynomial $\phi - \lambda s_j(\lambda)$ minimizes $\|\phi - \lambda s(\lambda)\|_w$ among all polynomials s of degree $\leq j-1$.

It is worth mentioning that there is an alternative formula for α_j which is

$$\alpha_j = \tilde{\alpha}_j - \frac{\langle 1-\phi, \lambda\pi_j\rangle}{\langle \lambda\pi_j, \lambda\pi_j\rangle} \ , \tag{24}$$

whose merit, relative to the expression used in Line 4 of the algorithm, is that it clearly establishes the new algorithm as a corrected version of the generic CR algorithm of Figure 8. In the special situation when $\phi \equiv 1$, $\alpha_i = \tilde{\alpha}_i$, and the two algorithms coincide as expected.

4.2 The base filter function

The solutions computed by the algorithms just seen consist of generating polynomial approximations to a certain base filter function ϕ. It is generally not a good idea to use as ϕ the step function because this function is discontinuous and approximations to it by high degree polynomials will exhibit very wide oscillations near the discontinuities. It is preferable to take as a "base" filter, i.e., the filter which is ultimately approximated by polynomials, a smooth function such as the one illustrated in Figure 10.

The filter function in Figure 10 can be a piecewise polynomial consisting of two pieces: A function which increases from zero to one when λ increases smoothly from 0 to γ, and the constant function unity in the interval $[\gamma, \ \beta]$. Alternatively, the function can begin with the value zero in the interval $[0, \ \gamma_1]$, then increase smoothly from 0 to one in a second interval $[\gamma_1, \ \gamma_2]$, and finally take the value one in $[\gamma_2, \ \beta]$. This second part of the function (the first part for the first scenario) bridges the values one and one by a smooth function and was termed a "bridge function" in [8].

Filtered Conjugate Residual Polynomials Algorithm	
Compute $\tilde{r}_0 := b - Ax_0$, $p_0 := \tilde{r}_0$	$\pi_0 = \tilde{\rho}_0 = 1$
	Compute $\lambda\pi_0$
for $j = 0, 1, \ldots,$ until convergence :	
$\quad\tilde{\alpha}_j := < \tilde{\rho}_j, \lambda\tilde{\rho}_j >_w \, / < \lambda\pi_j, \lambda\pi_j >_w$	
$\quad\alpha_j := < \phi, \lambda\pi_j >_w \, / < \lambda\pi_j, \lambda\pi_j >_w$	
$\quad x_{j+1} := x_j + \alpha_j p_j$	
$\quad\tilde{r}_{j+1} := \tilde{r}_j - \tilde{\alpha}_j A p_j$	$\tilde{\rho}_{j+1} = \tilde{\rho}_j - \tilde{\alpha}_j \lambda\pi_j$
$\quad\beta_j := < \tilde{\rho}_{j+1}, \lambda\tilde{\rho}_{j+1} >_w \, / < \tilde{\rho}_j, \lambda\tilde{\rho}_j >_w$	
$\quad p_{j+1} := r_{j+1} + \beta_j p_j$	$\pi_{j+1} := \tilde{\rho}_{j+1} + \beta_j \pi_j$
	Compute $\lambda\pi_{j+1}$
end	

Figure 9: The filtered Conjugate Residual polynomials algorithm

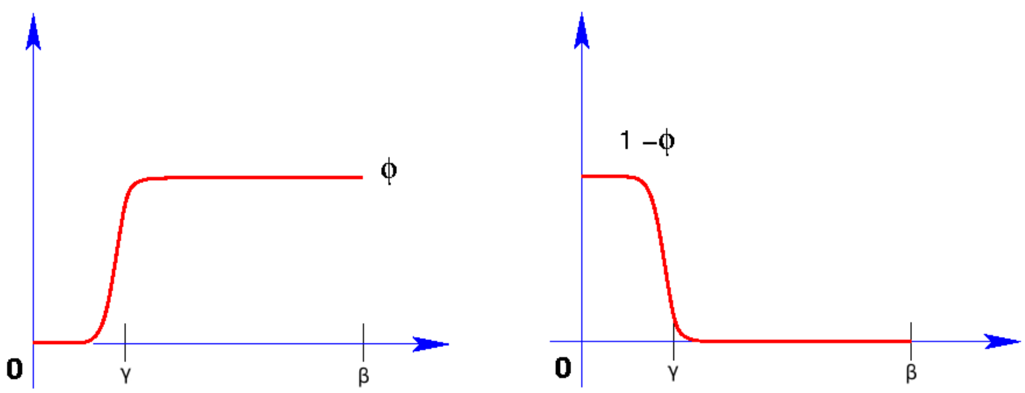

Figure 10: A typical filter function ϕ and its dual filter $1 - \phi$

A systematic way of generating base filter functions is to use bridge functions obtained from Hermite interpolation. The bridge function is an interpolating polynomial (in the Hermite sense) depending on two integer parameters m_0, m_1, and denoted by $\Theta_{[m_0,m_1]}$ which satisfies the following conditions:

$$\begin{array}{lll} \Theta_{[m_0,m_1]}(0) = 0; & \Theta'_{[m_0,m_1]}(0) = \cdots = & \Theta^{(m_0)}_{[m_0,m_1]}(0) = 0 \\ \Theta_{[m_0,m_1]}(\gamma) = 1; & \Theta'_{[m_0,m_1]}(\gamma) = \cdots = & \Theta^{(m_1)}_{[m_0,m_1]}(\gamma) = 0 \end{array} \tag{25}$$

Thus, $\Theta_{[m_0,m_1]}$ has degree m_0+m_1+1 and m_0, m_1 define the degree of smoothness at the points 0 and α respectively. The ratio $\frac{m_1}{m_0}$ determines the localization of the inflexion point. Making the polynomial increase rapidly from 0 to 1 in a small interval, can be achieved by taking high degree polynomials but this has the effect of slowing down convergence toward the desired filter as it has the effect of causing undesired oscillations. Two examples are shown in Figures 11 and 12.

Once the base filter is selected, the filtered CR algorithm can be executed. There remains however to define the inner products. Details on the weight functions and the actual techniques for computing inner products of polynomials can be found in [30]. We only mention that it is possible to avoid numerical integration by defining the inner products by using classical weights (e.g., Chebyshev) in each sub-interval of the whole interval where the base filter is defined. Since the base filter is a standard polynomial in each of these sub-intervals, inner products in these intervals can be evaluated without numerical integration. This, in effect, is equivalent to using Gaussian quadrature in each of these sub-intervals.

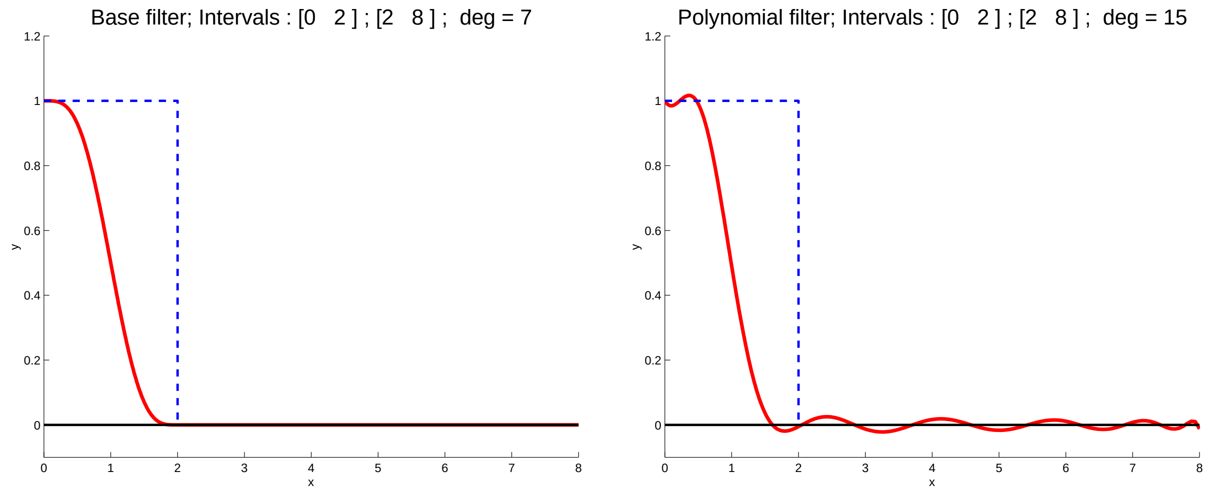

Figure 11: The base filter $\Theta_{[4,4]}$ in [0, 2] and one in [2, 8] and its polynomial approximation of degree 15.

There are a number of parameters which can be exploited to yield a desired filter polynommial. In addition to the degrees of the polynomials m_0, m_1, one can also define the weight functions differently. For example, more or less emphasis can be placed in each sub-interval. For now our codes use an equal weighting for each sub-interval. A forthcoming report [3] will document various aspects related to the selection of filter polynomials and other implementation issues.

5 Numerical Experiments

This section reports on a few numerical experiments with matrices taken from electronic structure calculations and from the Harwell-Boeing collection. Two good other references points for a useful comparison would be both the partially reorthogonalized Lanczos (which was used in [4]) and the implicitly restarted Lanczos iteration as it is implemented in the popular package `ARPACK` [16, 36]. We compare these two algorithms with our current implementation of the the Filtered Lanczos (`FLAN`) algorithm with partial reorthogonalization. We should begin by mentioning that we are aware of a number of possible improvements to the current implementation of `FLAN` which can be viewed as preliminary in nature.

All the experiments which follow have been performed on `SGI Origin 2000` system using a single `R12000` processor 300 `MHz`. The Filtered Lanczos code [3] is available from the authors upon request. `FLAN` is implemented purely in `C` while `ARPACK` is implemented in `Fortran 77`. The Lanczos algorithm with partial reorthogonalization is based in the `Fortran 77` code `PLANSO` [39]. The convergence tolerance was set to 10^{-10} for all methods. For `ARPACK` the maximum dimension of the Lanczos basis was always set equal to twice the number of requested eigenvalues.

In implicitly restarted techniques, such as the ones implemented in `ARPACK`, a basis of length equal to the number of required eigenvalues is updated at each restart. Thus, such methods are not designed to compute all eigenvalues in a given interval. This, of course, is in contrast to the Filtered Lanczos iteration, as well as to the unrestarted Lanczos algorithm. In order to facilitate a performance comparison we have used the following setting: for each test matrix we are interested in a given number of its algebraically smallest eigenvalues. We compute these using `ARPACK`. Then, we use the Filtered Lanczos and the unrestarted Lanczos iteration with partial reorthogonalization to compute all eigenvalues that are smaller or equal to the largest of the requested eigenvalues computed by `ARPACK`. Of course, this comparison is not carried out on completely equal terms. However, our goal is to demonstrate that a strategy of exchanging memory accesses with additional matrix vector products can significantly lower the overall computational

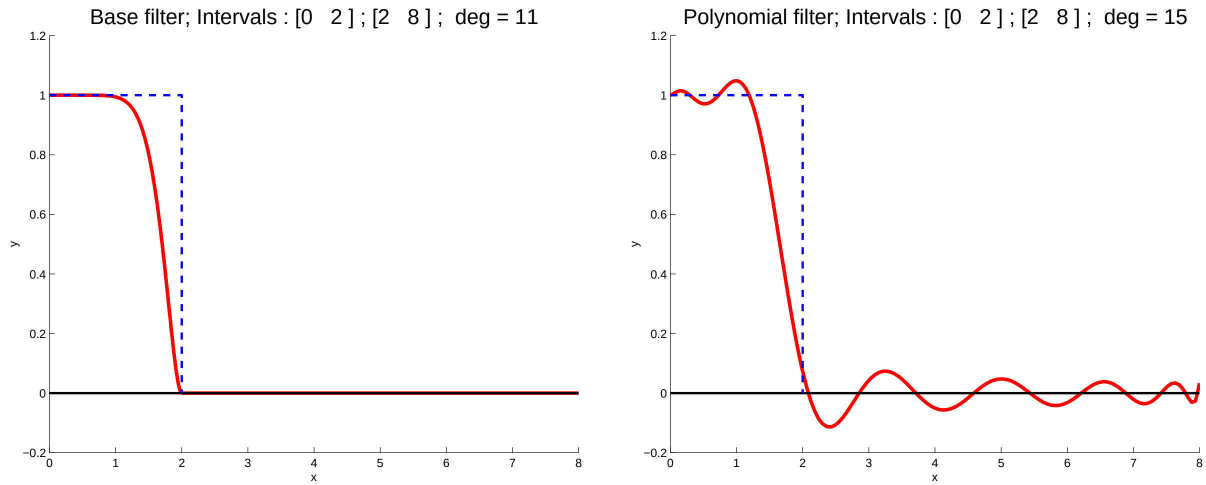

Figure 12: The base filter $\Theta_{[10,2]}$ in [0, 2] and one in [2, 8] and its polynomial approximation of degree 15.

matrix	size n	nnz	nnz/n
$\mathtt{Si_{10}H_{16}}$	17077	875923	51.3
$\mathtt{Ge_{99}H_{100}}$	94341	6332795	67.2
$\mathtt{Ge_{87}H_{76}}$	94341	5963003	63.2
$\mathtt{Si_{34}H_{36}}$	97569	5156379	52.8
`Andrews`	60000	760154	12.7

Table 1: Characteristics of test matrices: nnz is the total number of nonzeros, so the last column shows the average number of nonzeros per row.

cost. This was previously shown in [4], however at the important expense of additional memory, relative to implicitly restarted techniques. In this paper we show that the Filtered Lanczos iteration can achieve both goals: it can operate on limited memory while significantly reducing the overall computational cost.

Test matrices. We have used four matrices from electronic structure calculations for the tests. These are Hamiltonians obtained from a real space code [5]. In addition, we have also used a test matrix, namely the Andrews matrix, from the University of Florida sparse matrix collection[3]. Table 1 provides the characteristics of the test matrices. For the Hamiltonians the number of the requested eigenvalues generally correspond to physical properties of the corresponding molecular system. For example, $\mathtt{Si_{10}H_{16}}$ has 28 occupied states, while $\mathtt{Si_{34}H_{36}}$ has 86, $\mathtt{Ge_{87}H_{76}}$ has 212 and $\mathtt{Ge_{99}H_{100}}$ has 248. In order to test the scalability of the methods under study we requested additional eigenvalues as well. For the matrix `Andrews` we arbitrarily requested 100-400 eigenvalues. We point out that all statistics for the Filtered Lanczos algorithm include an initial call to the unrestarted Lanczos algorithm, with partial reorthogonalization, in order to approximate (upper and lower) bounds for the extremal eigenvalues.

Discussion. The experimental results clearly illustrate that the Filtered Lanczos algorithm achieves significant improvements over the other two competing methods. The performance improvement becomes more evident as the number of requested eigenvalues increases.

[3]http://www.cise.ufl.edu/research/sparse/

All of our test matrices are sparse. However, the degree of sparsity (as measured by the average number of nonzeros per row shown in the last column of Table 1) differs significantly between the "denser" $Ge_{99}H_{100}$ Hamiltonian and the "sparser" Andrews matrix. A careful look in the results illustrated in Table 2 clearly suggests that the improvements in run-times of the Filtered Lanczos algorithm over ARPACK is more pronounced for the sparser test matrices. Thus, although the number of matrix-vector products in the Filtered Lanczos algorithm increases relative to ARPACK, a significant gain results from avoiding to update a large number of eigenvectors, which standard methods do at every step.

In comparison with the unrestarted partially reorthogonalized Lanczos procedure observe that the Filtered Lanczos method always requires far less memory. In fact, the amount of additional memory in comparison to ARPACK is quite modest. Typically, the new method will require a Lanczos basis with length close to three times the number of computed eigenvalues. We also observe that for rather dense matrices and small number of eigenvalues (i.e. $Si_{34}H_{36}$ and $Si_{10}H_{16}$) the unrestarted Lanczos method with partial reorthogonalization is the fastest of the three methods. However, when a large invariant subspace is sought, then the unrestarted Lanczos method will tend to require a long basis, ultimately causing even infrequent reorthogonalizations to dominate the cost.

6 Conclusions

This paper presented a Filtered Lanczos iteration method (FLAN) for computing large invariant subspaces associated with the algebraically smallest eigenvalues of very large and sparse matrices. In contrast to restarted techniques (e.g. ARPACK) which repeatedly update a fixed number of basis vectors, FLAN is allowed to augment the search subspace until all eigenvalues smaller than a predetermined upper bound have converged. The loss of orthogonality of the Lanczos basis vectors is treated by a partial reorhtogonalization scheme [33]. One technique which FLAN and explicit/implicit restarted Krylov subspace algorithms have in common is the use of filtering polynomials, designed to dampen eigencomponents along "unwanted" parts of the spectrum. However, while restarted techniques apply these polynomials periodically (i.e. at each restart), the FLAN procedure applies a fixed, pre-computed, low-degree polynomial of A to the working Lanczos vector, which amounts to a polynomial preconditioning technique applied to A. We showed that if the unwanted eigendirections are thoroughly filtered from the starting vector of the Lanczos algorithm, then the application of the aforementioned small degree polynomial successfully prevents the unwanted directions from reappearing into the iteration, thus significantly expediting convergence towards the desired invariant subspace. Earlier work (see, e.g., [30]) showed how one can design a Conjugate Residual type iteration that efficiently applies a low pass filter in order to solve regularized linear systems. The low degree polynomial which is involved in this procedure is used in FLAN.

Experimental evidence clearly shows that FLAN achieves significant performance improvements over the most sophisticated restarted technique (i.e. ARPACK) while at the same time incurring very modest additional memory requirements. These gains in efficiency are obtained by essentially trading the repeated and costly updates of the working eigenbasis which is inherit in restarted techniques for additional matrix-vector products. Thus, the method will work quite well whenever matrix-vector products are not expensive.

Finally, we should point out that the filtering methods described here can be easily extended to the cases where the desired spectrum corresponds to the algebraically largest eigenvalues, and perhaps more importantly, to all eigenvalues in a given interval well inside the spectrum. This will be the subject of the forthcoming report [3].

Acknowledgments This work would not have been possible without the availability of excellent source codes for diagonalization. Specifically, our experiments made use of the PLAN code developed by Wu and Simon [39] and the ARPACK code of Lehoucq, Sorensen, and Yang [16].

Andrews												
	F. Lanczos				Partial Lanczos				ARPACK			
n_o	MV	RTH	MEM	t	MV	RTH	MEM	t	MV	RES	MEM	t
100	3320 (290)	130	133	330	1390	111	636	530	1616	24	92	2000
200	6110 (600)	186	275	803	2360	213	1080	1633	2769	21	183	6682
300	8270 (840)	224	385	1364	3120	298	1428	2976	3775	19	275	13572
400	10610 (1100)	267	504	2274	3970	393	1817	4997	4978	19	366	23762

$Si_{10}H_{16}$												
	F. Lanczos				Partial Lanczos				ARPACK			
n_o	MV	RTH	MEM	t	MV	RTH	MEM	t	MV	RES	MEM	t
28	1144 (100)	21	13	48	539	16	72	24	592	27	7.5	58
50	1864 (180)	40	23	86	930	35	124	61	1039	31	13.3	187
150	4384 (460)	86	60	244	1940	97	259	273	2129	21	40	1111
200	5284 (560)	88	73	315	2190	114	292	360	2676	20	53	1847

$Si_{34}H_{36}$												
	F. Lanczos				Partial Lanczos				ARPACK			
n_o	MV	RTH	MEM	t	MV	RTH	MEM	t	MV	RES	MEM	t
86	2317 (230)	42	171	778	1440	36	1098	605	1537	24	131	2877
100	3127 (320)	54	238	1105	1810	50	1380	907	2164	32	152	4800
150	4657 (490)	102	365	1799	2880	96	2195	2191	3085	32	229	9993
200	6007 (640)	134	476	2496	3580	129	2729	3431	3803	30	305	16099

$Ge_{87}H_{76}$												
	F. Lanczos				Partial Lanczos				ARPACK			
n_o	MV	RTH	MEM	t	MV	RTH	MEM	t	MV	RES	MEM	t
212	4476 (470)	88	338	1895	2710	88	1951	1993	2867	20	306	12145
300	8256 (890)	172	641	4130	4010	153	2887	4448	4673	25	432	28359
424	11406 (1240)	240	893	6624	5740	252	4132	9804	6059	23	611	51118

$Ge_{99}H_{100}$												
	F. Lanczos				Partial Lanczos				ARPACK			
n_o	MV	RTH	MEM	t	MV	RTH	MEM	t	MV	RES	MEM	t
248	5194 (550)	102	396	2379	3150	109	2268	2746	3342	20	357	16454
350	8794 (950)	178	684	4648	4570	184	3289	5982	5283	24	504	37371
496	12934 (1410)	270	1015	8374	6550	302	4715	13714	6836	22	714	67020

Table 2: Summary of experimental results for all 5 test matrices. MV denotes the total number of matrix-vector products, which for the Lanczos algorithm with partial reorthogonalization is also the dimension of the Lanczos basis used. For the Filtered Lanczos algorithm, the numbers in parentheses in the MV column denote the dimension of the Lanczos basis. RTH denotes the number of reorthogonalization steps. RES is the number of restarts for ARPACK. MEM denotes the required memory in Mbytes and t is the total time in secs. Finally, n_o is the number of requested eigenvalues.

References

[1] J. Baglama, D. Calvetti, and L. Reichel. IRBL: An Implicitly Restarted Block Lanczos Method for large-scale Hermitian eigenproblems. *SIAM J. Sci. Comput.*, 24(5):1650–1677, 2003.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, 2000.

[3] C. Bekas, E. Kokiopoulou, and Y. Saad. `FLAN`: A filtered Lanczos iteration code for very large symmetric eigenproblems. *Technical report under preparation*, 2005.

[4] C. Bekas, Y. Saad, M. Tiago, and J. Chelikowsky. Computing charge densities with partially reorthogonalized lanczos. *To appear in Comput. Phys. Comm.*, 2005.

[5] J. R. Chelikowsky, L. Kronik, I. Vasiliev, M. Jain, and Y. Saad. *Using Real Space Pseudopotentials for Electronic Structure Calculations*, volume 10 of *Handbook of Numerical Analysis, Special Volume on Computational Chemistry, (C. Le Bris, Guest editor)*, pages 613–637. Elsevier Science B. V., 2003.

[6] J. Cullum and R. A. Willoughby. *Lanczos algorithms for large symmetric eigenvalue computations.* Volumes 1 and 2. Birkhäuser, Boston, 1985.

[7] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, 1999.

[8] J. Erhel, F. Guyomarc, and Y. Saad. Least-squares polynomial filters for ill-conditioned linear systems. Technical Report umsi-2001-32, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2001.

[9] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 3d edition, 1996.

[10] Z. Jia and D. Niu. An implicitly restarted refined bidiagonalization lanczos method for computing a partial singular value decomposition. *SIAM J. Matrix. Anal. Appl.*, 25(1):246–265, 2003.

[11] S. Kaniel. Estimate for some computational techniques in linear algebra. *Math. Comp.*, 20, 1966. Lanczos bounds by kaniel.

[12] E. Kokiopoulou and Y. Saad. Polynomial Filtering in Latent Semantic Indexing for Information Retrieval. In *ACM-SIGIR Conference on research and development in information retrieval*, Sheffield, UK, July 25th-29th 2004.

[13] G. Kresse and J. J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 54(16):11169–11186, 1996.

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

[15] R. M. Larsen. *Efficient Algorithms for Helioseismic Inversion.* PhD thesis, Dept. Computer Science, University of Aarhus, DK-8000 Aarhus C, Denmark, October 1998.

[16] R. Lehoucq, D. C. Sorensen, and C. Yang. *Arpack User's Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restarted Arnoldi Methods.* SIAM, Philadelphia, 1998. URL `http://www.caam.rice.edu/software/ARPACK`.

[17] J. L. Martins and M. Cohen. Diagonalization of large matrices in pseudopotential band-structure calculations: Dual-space formalism. *Physical Review B*, 37(11):6134–6138, 1988.

[18] J. L. Martins, N. Troullier, and S.-H. Wei. Pseudopotential plane-wave calculations for ZnS. *Physical Review B*, 43(3):2213–2217, 1991.

[19] F. Mauri and G. Galli. Electronic-structure calculations and molecular-dynamics simulation with linear system size scaling. *Physical Review B*, 50(7):4316–4326, 1994.

[20] F. Mauri, G. Galli, and R. Car. Orbital formulation for electronic-structure calculations with with linear system size scaling. *Physical Review B*, 47(15):9973–9976, 1993.

[21] A. Neumaier. Solving ill-conditioned and singular linear systems : a tutorial on regularization. *Siam Rev.*, 40(3):636–666, September 1998.

[22] C. C. Paige. *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*. PhD thesis, London University, London, England, 1971.

[23] C.C. Paige, B.N. Parlett, and H.A. van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Num. Lin. Algebra with Appl.*, 2(2):115–134, 1995.

[24] B. N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM, 1998.

[25] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos. Iterative minimization techniques for ab-initio total energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.*, 64:1045–1097, 1992.

[26] H. Rutishauser. Simultaneous iteration for symmetric matrices. In J.H. Wilkinson and C.H. Reinsch, editors, *Handbook for automatic computations (linear algebra)*, pages 202–211, New York, 1971. Springer Verlag.

[27] Y. Saad. On the rates of convergence of the Lanczos and block Lanczos methods. *SIAM J. Numer. Anal.*, 17(5):687–706, October 1980.

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

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

[30] Y. Saad. Filtered conjugate residual-type algorithms with applications. Technical Report umsi-2005-042, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2005.

[31] Y. Saad, A. Stathopoulos, J. Chelikowsky, K. Wu, , and S. Öğüt. Solution of large eigenvalue problems in electronic structure calculations. *BIT*, 36(3):563–578, 1996.

[32] A. H. Sameh and J. A. Wisniewski. A trace minimization algorithm for the generalized eigenvalue problem. *SIAM Journal on Numerical Analysis*, 19:1243–1259, 1982.

[33] H. D. Simon. Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear Algebra Appl.*, 61:101–132, 1984.

[34] H. D. Simon. The Lanczos algorithm with partial reorthogonalization. *Math. Comp.*, 42(165):115–142, 1984.

[35] G. L. G. Sleijpen and H. A. van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Rev.*, 42(2):267–293, 2000.

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

[37] G. W. Stewart. Adjusting the rayleigh quotient in semiorthogonal Lanczos methods. *SIAM J. Scient. Comput.*, 24(1):201–207, 2002. Stewart semiorthogonal.

[38] J. H. Wilkinson and C. Reinsch. *Handbook for automatic computation, Vol. II, Linear Algebra.* Springer Verlag, New York, 1971.

[39] K. Wu and H. Simon. A parallel Lanczos method for symmetric generalized eigenvalue problems. Technical Report 41284, Lawrence Berkeley National Laboratory, 1997. Available on line at `http://www.nersc.gov/research/SIMON/planso.html`.