© 2008 Society for Industrial and Applied Mathematics

COMPUTATION OF LARGE INVARIANT SUBSPACES USING POLYNOMIAL FILTERED LANCZOS ITERATIONS WITH APPLICATIONS IN DENSITY FUNCTIONAL THEORY[*]

C. BEKAS[†], E. KOKIOPOULOU[‡], AND YOUSEF SAAD[§]

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 planewave 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 symmetric/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 technique 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.

1. Introduction and preliminaries. Ab initio electronic structure calculations, in the framework of density functional theory (DFT) [8, 11], have proven remarkably accurate in providing a wealth of information concerning several important physical properties of complex materials. However, DFT calculations are extremely demanding and have stretched our computational capabilities to their very limits. Therefore, advances in better simulation techniques and algorithms receive much attention in this very active field of research.

The core problem in DFT calculations is the solution of the time-independent Schrödinger equation

$$\mathcal{A}_\varrho \Psi_\varrho = E \Psi_\varrho, \tag{1}$$

[†]Current address: IBM, Zurich Research Laboratory, Säumerstrasse 4, CH-8803, Rüschlikon, Switzerland (bek@zurich.ibm.com).

[‡]Current address: EPFL, Signal Processing Institute-ITS, CH-1015, Lausanne, Switzerland (effrosyni.kokiopoulou@epfl.ch).

[§]Computer Science & Engineering Department, University of Minnesota, Twin Cities, 200 Union St. SE, Minneapolis, MN 55455 (saad@cs.umn.edu).

where ϱ is the charge density of the electrons distribution, $\mathcal{A}_\varrho$ is the Hamiltonian operator, Ψ_ϱ are the wavefunctions, and E is the energy of the system. Observe that this is a nonlinear eigenvalue problem, since the Hamiltonian and the wavefunctions depend upon each other through the charge density ϱ. Recent decades have seen many methods that attempt to efficiently solve (1). All of them utilize some sort of iteration which aims to improve some initially selected wavefunctions so that at the end of the iteration the approximate energy E is as small as possible, or in other words the solution of (1) is self-consistent.

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

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

where the summation is taken over all occupied states (valence electrons) n_o of the system under study. This is a crucial calculation in DFT since the potential V of the Hamiltonian $\mathcal{A} = \nabla^2 + V$ depends on the charge density ϱ, which in turn depends on the eigenvectors Ψ_i of $\mathcal{A}$ (see (2)), and, as a result, an iterative loop is required to achieve self-consistence. Computing the charge density $\varrho(r)$ via (2) 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 of the Hamiltonian. This is because $\varrho(r)$ is invariant under 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"

$$(3) \qquad P = Q_{n_o} Q_{n_o}^\top \quad \text{with} \quad Q_{n_o} = [\psi_1, \ldots, \psi_{n_o}].$$

Specifically, the charge density at the jth point r_j is the jth 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 led to improved schemes which do not focus on extracting individual eigenvectors. For example, [1] showed that the semiorthogonal 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 and the accurate calculations of the eigenvectors. When using standard diagonalization software, much attention is paid to obtaining accurate eigenvectors, at a cost that is often quite high. If one focuses on invariant subspaces, all that is needed is that a good basis of the subspace be computed, but this basis does not need to be a basis of accurate eigenvectors. For example, a set of m vectors which are linearly independent and which are known to have no components in the undesired eigenvectors will constitute such a basis, and an orthonormal basis can be obtained from it if we want to compute the charge density ϱ. Approximate eigenvectors can be extracted from this basis (by a Rayleigh–Ritz projection process) but this is not necessary. Shifting the focus from individual eigenvectors to bases of invariant subspaces can reduce the cost considerably.

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 $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, 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 $\lambda_1, \ldots, \lambda_{n_o}$. We assume that we are given an interval $[\alpha, \beta]$ which (tightly) contains the spectrum of A. The nature of the algorithms used in this paper 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.

When the number of desired eigenvalues is rather small, say on the order of a few dozen, the problem can be addressed by a number of successful algorithms. Among these is an extensively used general purpose method based on implicitly restarted Lanczos iterations [33], and implemented in the software package ARPACK [16]. However, the problem becomes much harder in the case when we seek to compute the invariant subspace associated with 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 typically on the order of a few hundred thousand to several million.

The method proposed in this paper exploits two distinct and complementary tools to address 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 technique employs a well-selected low pass filter polynomial, obtained via a conjugate residual- (CR-)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.

Earlier papers presented these two tools in the literature. For example, the filter polynomial used here is borrowed from [28], and earlier variants were used in [12] and [6]. The use of the partial reorthogonalization Lanczos (PR-Lanczos) was suggested in [1]. However, one of the difficulties with the method in [1] is that very large bases are often required. Thus, the goal of the present paper is to show how to effectively combine these two distinct and powerful tools, namely, polynomial filtering on the one hand and PR-Lanczos on the other, to solve the difficult problem of extracting large invariant subspaces. The motivation for using polynomial filtering in various applications, including computing large invariant subspaces, was also discussed in [28].

1.1. Previous work. An alternative viewpoint which appears in existing DFT codes is to replace diagonalization by "direct minimization," which in effect amounts to computing the subspace of minimum trace, i.e., an orthogonal basis $Q = [q_1, \ldots, q_{n_0}]$ such that $\operatorname{tr}(Q^T A Q)$ is minimum. In fact, many publications of the mid 1990s focused on avoiding orthogonality, which turned out to be hard to achieve. A method that was explicitly based on "trace-minimization" was proposed by Sameh and Wisniewski [30] 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 ith 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.

Other 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]. For example, preconditioners for eigenvalue problems in which planewaves are used can be easily extracted from matrices which use lower-dimensional representations of the Hamiltonian, i.e., Hamiltonians obtained from using fewer planewaves and extended to higher dimensions in some simple way. The lower-dimensional Hamiltonians have good representatives of the desired eigenvectors of the higher-dimensional ones, and this class of preconditioners can be quite effective in this context. In real-space methods, the situation is quite different. In this case, we found that preconditioning the eigenvalue problem is much harder [29]. Generally the gains with the standard preconditioners that were attempted were small, and these are outweighed by the additional cost of applying the preconditioner and by 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, but this is no longer possible with the Davidson approach, which requires a full orthogonalization at each step. In [1] we explored this approach. The Lanczos algorithm was adapted in a number of ways, the most important of which was to replace the reorthogonalization step by a partial reorthogonalization scheme [15, 24, 31, 32].

The use of matrix polynomials and filtering has been used in other ways, and the idea has played a prominent role in linear scaling and related methods; see, for example, [9, 10, 17, 21, 22]. In some cases, these methods will consist of computing the entire density matrix (3) [21] or a small part of it as an approximation [10].

1.2. The Lanczos procedure. The Lanczos algorithm [14] (see also [3, 4, 7, 24, 26]) builds a sequence of vectors $q_1, q_2, \ldots, q_m$ which form an orthonormal basis $Q_m \in \mathbb{R}^{n \times m}$ of the Krylov subspace

$$(4) \qquad \mathcal{K}_m(A, q_1) = \mathtt{span}\{q_1, Aq_1, A^2q_1, \ldots, A^{m-1}q_1\},$$

where q_1 is an arbitrary (typically random) initial vector with $\|q_1\| = 1$. As is well known, this sequence of vectors satisfies the 3-term recurrence

$$(5) \qquad \beta_{i+1}q_{i+1} = Aq_i - \alpha_i q_i - \beta_i q_{i-1}.$$

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.

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

$$(6) \qquad T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{bmatrix},$$

where the scalars α_i, β_i are computed by the Lanczos algorithm, then it can be verified that $AQ_m = Q_m T_m + \beta_{m+1}q_{m+1}e_m^\top$, where e_m is the mth column of the canonical basis and q_{m+1} is the last vector computed by the Lanczos algorithm.

The eigenvalues of matrix A are approximated by those of matrix T_m. 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].

2. Computing large eigenspaces with the Lanczos procedure. In the situation when large eigenspaces are to be computed by the Lanczos algorithm, the number m of Lanczos steps required for all the desired eigenvectors to converge can be quite large. Therefore, if the algorithm is to be applied without any form of restarting or preconditioning, then we will have to deal with two related demands: (1) the need to apply some form of reorthogonalization to the Lanczos vectors [1, 15, 24, 31, 32], and (2) the need to store the Lanczos basis Q_m because it is needed by the reorthogonalization steps. The 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 from 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 $O(m^2 n)$ 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., [5, 13, 18, 19, 36].

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 [1] 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 2000 vectors of length 1 million requires "only" 16 GB of memory, which is certainly within reach of most high-performance computers.[1] However, for larger calculations this will be an enormous burden and out-of-core algorithms would be needed.

2.1. Use of partial reorthogonalization. A remarkable property of the Lanczos algorithm is that, in theory (exact arithmetic), it computes a basis of the Krylov subspace, which is *orthonormal*. This is done with a simple 3-term recurrence. However, in practice, i.e., in the presence of finite precision arithmetic (e.g., double precision floating point arithmetic), the basis vectors quickly start to lose orthogonality. The onset of loss of orthogonality is sudden and takes place as soon as one or more eigenvectors start converging, as was discovered in the seminal work of Paige [23]. As soon as this happens, the orthogonality is completely lost very rapidly, indicating an unstable underlying computation. As an illustration, consider the Hamiltonian ($n = 17077$) corresponding to $Si_{10}H_{16}$, which was obtained by the real-space code PARSEC.[2] 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 1 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,

---

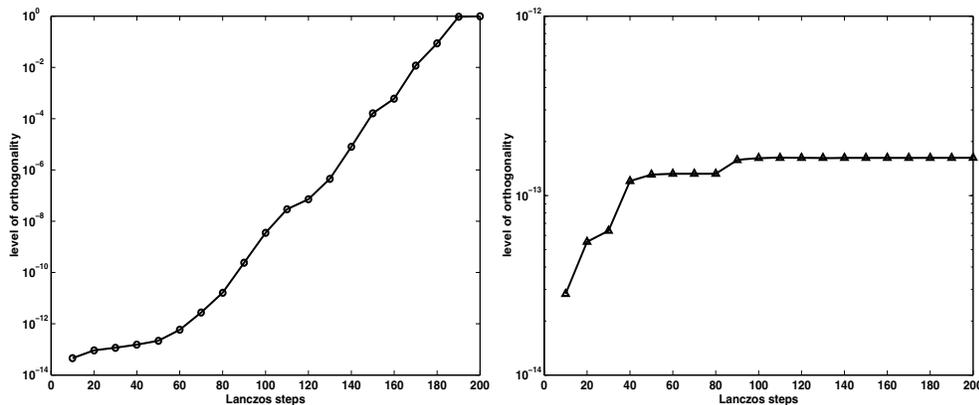[1]In modern high-performance computers this will typically be available in a single node.
[2]http://www.ices.utexas.edu/parsec/index.html

FIG. 1. *Levels of orthogonality of the Lanczos basis for the Hamiltonian ($n = 17077$) corresponding to $Si_{10}H_{16}$. Left: Lanczos without reorthogonalization. Right: Lanczos with partial reorthogonalization. The number of reorthogonalizations was 34 with an additional 3400 inner vector products.*

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 [34]). However, full reorthogonalization can be a costly procedure.

An alternative is *partial reorthogonalization*, which attempts to reorthogonalize only when it is deemed necessary. The goal is not so much to guarantee that the vectors are exactly orthogonal as 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 [31] 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, 32]. 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 [32]:

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

where the scalars α_i and β_i, $i = 1, \ldots,$ are identical to the ones computed by the Lanczos algorithm.

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 right plot in Figure 1 illustrates the corresponding level of orthogonality when partial reorthogonalization is applied. Only 34 reorthogonalization steps were required, compared with the 200 that would have been required if full reorthogonalization was employed.

It was shown in [1] 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.

2.2. Polynomial acceleration and restarting techniques. The above discussion strongly suggests that it is critical to use a Lanczos basis that is as small as possible. In order to achieve this, we can apply the Lanczos process not to the original matrix A but rather on a matrix $p(A)$, where $p(t)$ 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, [26] and the references therein). Typically, the goal is to restart the Lanczos procedure after a fixed number of iterations with a starting vector from which unwanted eigendirections have been filtered out. In this paper we follow a different approach. We do not employ any restarts, but rather filter each matrix-vector product in the Lanczos process using a small number of CR-type iterations with the matrix A. As can be expected, 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.

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 Krylov subspace is not allowed to grow indefinitely. When a maximum number of iterations $M_{\max}$ is reached, a new starting vector $q_1^{(k+1)}$ is selected and the process is restarted; see [26, 33] for details.

Whether explicit or implicit, 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. This is illustrated in Figure 2. The matrix that is tested corresponds to a second order finite difference 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 2 illustrates that at the first step of the Lanczos procedure, the vector q_1 is orthogonal (up to machine precision) to the unwanted eigenvectors. However, it takes only $m = 13$ 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 and assume that $\langle q_1, \psi_i \rangle = \epsilon$ for a given eigenvector ψ_i with $i > n_o$. Recall that the Lanczos vector q_{m+1} is of the form $q_{m+1} = z_m(A)q_1$, where z_m is a polynomial of degree m, called the $(m + 1)$st Lanczos polynomial. The sequence of polynomials z_k, $k = 1, \ldots, m$, is orthogonal with respect to a certain discrete inner product. Since the initial vector has very small components in the eigenvectors associated with eigenvalues $\lambda_i > \lambda_{n_o}$, it is to be expected that the Lanczos polynomial z_m is such
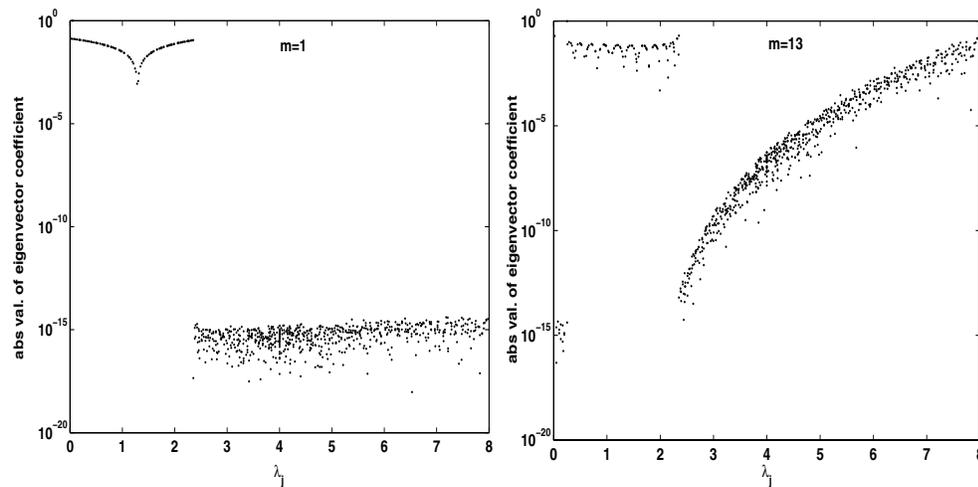
FIG. 2. *Coefficients of the last basis vector q_m of the Lanczos procedure (no partial reorthogonalization was required) for the discretization of the Laplacian, when the starting vector does not have any components in undesired eigenvectors. Left: one step. Right: $m = 13$ steps.*

that $z_m(\lambda_i) \gg 1$ for $i > n_o$. Therefore, we will have

$$
\begin{aligned}
\langle q_{m+1}, \psi_i \rangle &= \langle z_m(A)q_1, \psi_i \rangle \\
&= \langle q_1, z_m(A)\psi_i \rangle \\
&= z_m(\lambda_i)\langle q_1, \psi_i \rangle \\
(8) \qquad\qquad &= z_m(\lambda_i)\epsilon.
\end{aligned}
$$

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

The situation can be remedied by replacing A by an operator of the form $B = p(A)$, where $p(\lambda_i)$ 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.

Before we state the result in detail, we must recall that in inexact arithmetic, the Lanczos relation (5) is replaced by a relation of the form

$$
(9) \qquad\qquad Aq_i = \beta_{i+1}q_{i+1} + \alpha_i q_i + \beta_i q_{i-1} - z_i,
$$

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 $\delta \equiv p(\lambda)$. Assume that the sequence $\{q_i\}$ satisfies the model (9) and define $\epsilon_i^\psi = \langle \psi, z_i \rangle$. Then the scalar sequence $\sigma_i = \langle q_i, \psi \rangle$ satisfies the recurrence*

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

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

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

in which $\varepsilon_m = [\epsilon_1^\psi, \epsilon_2^\psi, \ldots, \epsilon_m^\psi]^\top$ *and* T_m *is the tridiagonal matrix* (6).

Proof. Let $B \equiv p(A)$. 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

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

Since $B\psi = \delta\psi$, this readily yields the expression (10).

Define the vector $s_m = [\sigma_1, \sigma_2, \ldots, \sigma_m]^\top$. We can rewrite the relations (10) 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 (11). □

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 (11). 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^ψ's 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 (11) becomes

$$\tag{12} \sigma_{m+1} = \frac{\hat{\varepsilon}_m^\top (T_m - \delta I)^{-1}e_1}{\beta_{m+1}e_m^\top (T_m - \delta I)^{-1}e_1}.$$

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 (8)), 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 a proper inner polynomial $p(t)$ 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 towards the corresponding eigenvector. By the time convergence takes place, the desirable subspace will have already been computed.
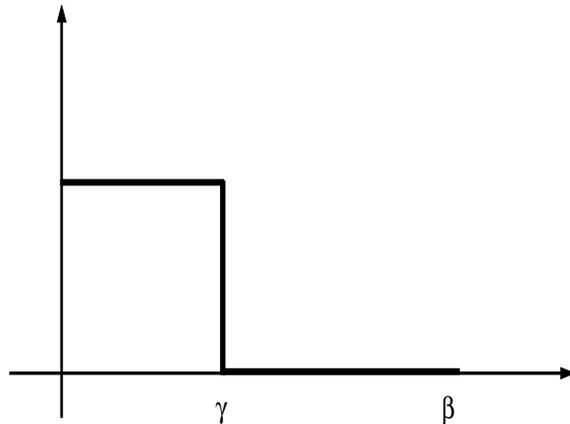
FIG. 3. *The Heaviside function for the interval* $[\gamma, \beta]$.

3. The filtered Lanczos procedure. Partial reorthogonalization can significantly extend the applicability of Lanczos in electronic structure calculations (see [1]), 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 λ_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 in the usual Lanczos algorithm by $p(A)q_i$, where $p(t)$ is a low-degree polynomial filter that approximates the Heaviside function (see Figure 3). The interval $[\gamma, \beta]$ contains all the unwanted (largest) eigenvalues, which are approximately mapped by $p(t)$ 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 $p(A)$ with the vector q_i. Let d be the degree of the polynomial $p(t)$. 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 the matrix $p(A)$ which is also an invariant subspace for A itself. However, while the restriction of $p(A)$ on the orthogonal Lanczos basis Q_m is a tridiagonal matrix, i.e., $Q_m^\top p(A) Q_m = T_m$ is tridiagonal, this is no longer true for A, i.e.,

$$(13) \qquad\qquad Q_m^\top A Q_m = \tilde{T}_m,$$

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 $p(A)$. However, A and $p(A)$ 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 $\hat{T}_m = Y^\top Q_m^\top A Q_m Y$.

Similarly to the Lanczos procedure, the basis vectors q_i in the filtered Lanczos

procedure are also expected to rapidly lose orthogonality. Thus, the partial reorthogonalization techniques of section 2.1 will prove to be particularly useful in the practical deployment of the method.

The larger the degree of the polynomial $p(t)$, 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 $p(t)$ to take small values on the region of the unwanted eigenvalues. Section 4 discusses a CR-type iteration that achieves this goal. In order to describe the filtered Lanczos iteration, it suffices for the time being to consider the application of the filtering polynomial as a "black box" function $\mathcal{P}(A, q_i, d)$.

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 the 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 p_h on a random vector $\tilde{q}$, such that $q_1 = p_h(A)\tilde{q}$. 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. A systematic way to stop this initial iteration is to monitor the norm of the residual of the CR iteration, which indicates how well the sequence of the orthogonal CR polynomials approximate the base filter function. Once this norm, which is cheap to compute (see section 4.1), falls below a user-specified tolerance, the iteration is stopped. This in turn will guarantee that unwanted large eigendirections have been adequately dampened.

Bounding intervals. 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 obtained from a few Lanczos steps. Then we use the practical bounds $\tilde{\alpha} = \tilde{\lambda}_1 - \|r_1\|$ and $\tilde{\beta} = \tilde{\lambda}_n + \|r_n\|$. If $\tilde{\alpha}$ is negative, then we shift the Hamiltonian so as to make all its eigenvalues positive. The "interval of wanted eigenvalues" is user-defined. Standard "self-consistent field" iterations in electronic structure methods are inherently nonlinear iterations and as such information from previous iterations can be exploited to obtain a good estimate for the wanted interval. At the beginning of the self-consistent field loop, there are adequate initializations based on superposition of atomic wavefunctions that can be exploited. The "unwanted" interval is readily defined from the above.

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

Convergence criterion. Equally important in limiting 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 $(\tilde{\lambda}_i, y_i)$ is an eigenpair of the dense matrix $\tilde{T}_m$ (13). Then it is natural to monitor the norm of the residual $r_i = A\tilde{x}_i - \tilde{\lambda}_i \tilde{x}_i$. 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. In order to reduce the computational cost as well as the memory load, we opt to avoid calculating the eigenvectors y_i at every step. Thus, we choose to monitor, during the iteration, the sum of the eigenvalues $\tilde{\lambda}_i$ of matrix $\tilde{T}_k$, which correspond to those eigenvalues of A that are smaller than the upper bound γ, $s_k = \sum_{\tilde{\lambda}_i < \gamma} \tilde{\lambda}_i$. Only when the change in the sum s_k, in comparison to s_{k-1}, is less than a user-defined tolerance do we calculate the eigenvectors y_i and thus check convergence by means of the residual norms $\|r_i\|$. If all residual norms are adequately small, then we stop the iteration. Otherwise, we continue the iteration and repeat the process. Further savings are achieved by not performing the convergence test for s_k at every Lanczos step, but only infrequently, for example, at fixed intervals.

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

$$(14) \qquad \tilde{T}_i = Q_{i+1}^\top A \, Q_{i+1} = [Q_i \quad q_{i+1}]^\top A [Q_i \quad q_{i+1}] = \begin{bmatrix} Q_i^\top A Q_i & Q_i^\top A q_{i+1} \\ q_{i+1}^\top A Q_i & q_{i+1}^\top A q_{i+1} \end{bmatrix}.$$

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` 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 is 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 4 shows a high-level algorithmic description of the filtered Lanczos iteration.

4. Polynomial filters. This section focuses on the problem of defining and applying the polynomial filter. Details on the algorithms described here can be found in [28]. We begin with a brief summary of filtering techniques when solving linear systems of equation by "regularization" [20]. 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 [28], computing a filtered solution amounts to computing a vector $s(A)b$ whose residual vector $p(A)b = b - As(A)b$ is a certain filter polynomial, typically one that is computed to be close to 1 for small eigenvalues and close to 0 for larger eigenvalues. In other words, it would resemble the desired filter polynomial, such as the one shown on the right of Figure 7.

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 is $p_{j+1}(\lambda) = 1 - \lambda s_j(\lambda)$. This polynomial is of degree $j + 1$ and has value 1 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,

FILTERED LANCZOS ALGORITHM.
(*Input*)
Matrix $A \in \mathbb{R}^{n \times n}$, starting vector q_1, $\|q_1\|_2 = 1$,
polynomial filter function $\mathcal{P}(A, q, d)$ that approximates the step function,
high polynomial degree d_h, stride $\mathtt{strd}$, 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

1.	Set $\beta_1 = 0$, $q_0 = 0$
2.	Thoroughly filter initial vector $q_1 = \mathcal{P}(A, q_1, d_h)$, $q_1 = q_1/\|q_1\|$
3.	**for** $i = 1, \ldots$
4.	$w_i = \mathcal{P}(A, q_i, d_l) - \beta_i q_{i-1}$
5.	$\alpha_i = \langle w_i, q_i \rangle$
6.	$w_i = w_i - \alpha_i q_i$
7.	$\beta_{i+1} = \|w_i\|_2$
8.	**if** $(\beta_{i+1} == 0)$ **then** stop
9.	$q_{i+1} = w_i/\beta_{i+1}$
10.	**if** $\mathtt{rem}(i, strd) == 0$ **then**
11.	Compute last row/column of matrix $\tilde{T}_i = Q_i^\top A Q_i$
11.	Compute all eigenvalues $\tilde{\lambda}_j$ of $\tilde{T}_i$ such that $\tilde{\lambda}_j < \gamma$
12.	Compute $s_i = \sum_{\tilde{\lambda}_i < \gamma} \tilde{\lambda}_i$
13.	**if** $(|(s_i - s_{i-1})/s_{i-1}| < \mathtt{tol})$ **then**
14.	Calculate residuals $\|r_i\|$ and if all $\|r_i\| < \mathtt{tol}$ **then break**
15.	**end**
16.	**end**

FIG. 4. *The filtered Lanczos algorithm. The inner product for vectors is denoted by $\langle ., . \rangle$.*

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 CR algorithm (see, e.g., [27]), 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

$$\langle p, q \rangle_g.$$

The related generic CR algorithm is given in Figure 5.

It can be easily shown that the residual polynomial p_j generated by this algorithm minimizes $\|p(\lambda)\|_g$ among all polynomials of the form $p(\lambda) = 1 - \lambda s(\lambda)$, where s is any polynomial of degree $\leq j - 1$. In other words, p_j minimizes $\|p(\lambda)\|_g$ among all

GENERIC CONJUGATE RESIDUAL ALGORITHM.
1.   Compute $r_0 := b - Ax_0$, $p_0 := r_0$, $\pi_0 = p_0 = 1$
2.   Compute $\lambda \pi_0$
3.   **for** $j = 0, 1, \ldots,$ until convergence:
4. $\alpha_j := \langle p_j, \lambda p_j \rangle_g / \langle \lambda \pi_j, \pi_j \rangle_g$
5. $x_{j+1} := x_j + \alpha_j p_j$
6. $r_{j+1} := r_j - \alpha_j A p_j$ $p_{j+1} = p_j - \alpha_j \lambda \pi_j$
7. $\beta_j := \langle p_{j+1}, \lambda p_{j+1} \rangle_g \langle p_j, \lambda p_j \rangle_g$
8. $p_{j+1} := r_{j+1} + \beta_j p_j$ $\pi_{j+1} := p_{j+1} + \beta_j \pi_j$
9.   Compute $\lambda \pi_{j+1}$
10.   **end**

FIG. 5. *Generic CR algorithm.*

polynomials p of degree $\leq j$, such that $p(0) = 1$. 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 $p_m(t)$ can be used as a filter polynomial for a given iteration. For example, the residual polynomial shown on the right of Figure 7, which is of the form $p(\lambda) = 1 - \lambda s(\lambda)$, can be used for computing all eigenvalues in the interval $[0, 1.7]$. The dual filter $1 - p(\lambda)$ has small values in $[0, 1.7]$ and 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 [6] 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 Figure 5). 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 $\tilde{p}_j$ 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 6, where now $\tilde{p}_j$ is the polynomial associated with the auxiliary sequence $\tilde{r}_j$.

FILTERED CONJUGATE RESIDUAL POLYNOMIALS ALGORITHM.

1.   Compute $\tilde{r}_0 := b - Ax_0$, $p_0 := \tilde{r}_0$ $\pi_0 = \tilde{p}_0 = 1$
2.   Compute $\lambda\pi_0$
3.   **for** $j = 0, 1, \ldots$, until convergence:
4. $\tilde{\alpha}_j := \langle \tilde{p}_j, \lambda\tilde{p}_j \rangle_w / \langle \lambda\pi_j, \lambda\pi_j \rangle_w$
5. $\alpha_j := \langle \phi, \lambda\pi_j \rangle_w / \langle \lambda\pi_j, \lambda\pi_j \rangle_w$
6. $x_{j+1} := x_j + \alpha_j p_j$
7. $\tilde{r}_{j+1} := \tilde{r}_j - \tilde{\alpha}_j Ap_j$ $\tilde{p}_{j+1} = \tilde{p}_j - \tilde{\alpha}_j \lambda\pi_j$
8. $\beta_j := \langle \tilde{p}_{j+1}, \lambda\tilde{p}_{j+1} \rangle_w / \langle \tilde{p}_j, \lambda\tilde{p}_j \rangle_w$
9. $p_{j+1} := r_{j+1} + \beta_j p_j$ $\pi_{j+1} := \tilde{p}_{j+1} + \beta_j \pi_j$
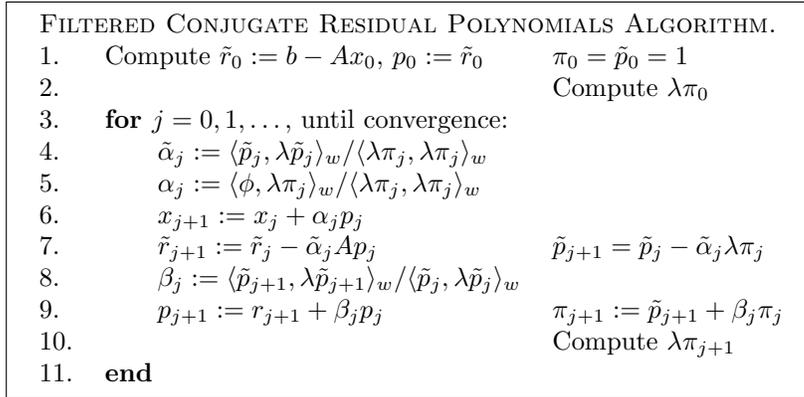10.   Compute $\lambda\pi_{j+1}$
11. **end**

FIG. 6. *The filtered CR polynomials algorithm.*

The only difference with a generic CR-type algorithm (see, e.g., Figure 5) 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 jth 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 jth degree polynomial:

$$(15) \qquad s_j(\lambda) = \alpha_0 \pi_0(\lambda) + \cdots + \alpha_j \pi_j(\lambda).$$

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

$$(16) \qquad \langle \lambda\pi_j(\lambda), \lambda\pi_i(\lambda) \rangle_w = \langle \lambda\tilde{p}_j(\lambda), \tilde{p}_i(\lambda) \rangle_w = 0 \quad \text{for} \quad i \neq j.$$

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

$$(17) \qquad \alpha_j = \tilde{\alpha}_j - \frac{\langle 1 - \phi, \lambda\pi_j \rangle}{\langle \lambda\pi_j, \lambda\pi_j \rangle},$$

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 5. 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 ϕ as 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 7.
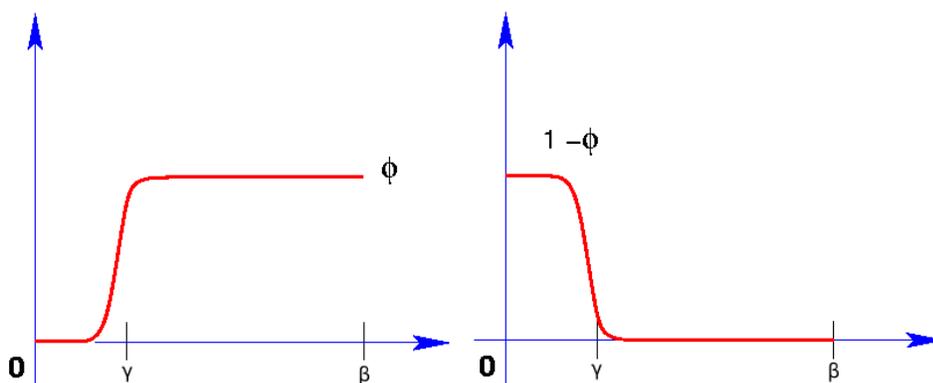
FIG. 7. *A typical filter function ϕ and its dual filter $1 - \phi$.*

The filter function in Figure 7 can be a piecewise polynomial consisting of two pieces: A function which increases from 0 to 1 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 0 in the interval $[0, \gamma_1]$, then increase smoothly from 0 to 1 in a second interval $[\gamma_1, \gamma_2]$, and finally take the value 1 in $[\gamma_2, \beta]$. This second part of the function (the first part for the first scenario) bridges the values 1 and 1 by a smooth function and was termed a "bridge function" in [6].

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:

$$
(18) \qquad
\begin{aligned}
\Theta_{[m_0, m_1]}(0) = 0; &\quad \Theta'_{[m_0, m_1]}(0) = \cdots = \quad \Theta^{(m_0)}_{[m_0, m_1]}(0) = 0, \\
\Theta_{[m_0, m_1]}(\gamma) = 1; &\quad \Theta'_{[m_0, m_1]}(\gamma) = \cdots = \quad \Theta^{(m_1)}_{[m_0, m_1]}(\gamma) = 0.
\end{aligned}
$$

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 inflection 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 causes undesired oscillations. Two examples are shown in Figures 8 and 9.

Once the base filter is selected, the filtered CR algorithm can be executed. It 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 [28]. 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 subinterval of the whole interval where the base filter is defined. Since the base filter is a standard polynomial in each of these subintervals, inner products in these intervals can be evaluated without numerical integration. This, in effect, is equivalent to using Gaussian quadrature in each of these subintervals.

The support of the bridge function, an interval in which the base function drops from 1 to 0, can be determined by the interval of wanted eigenvalues and the largest eigenvalue of the matrix. We already discussed how to get the required bounds for the largest and smallest eigenvalues of A from a few steps of the Lanczos algorithm.
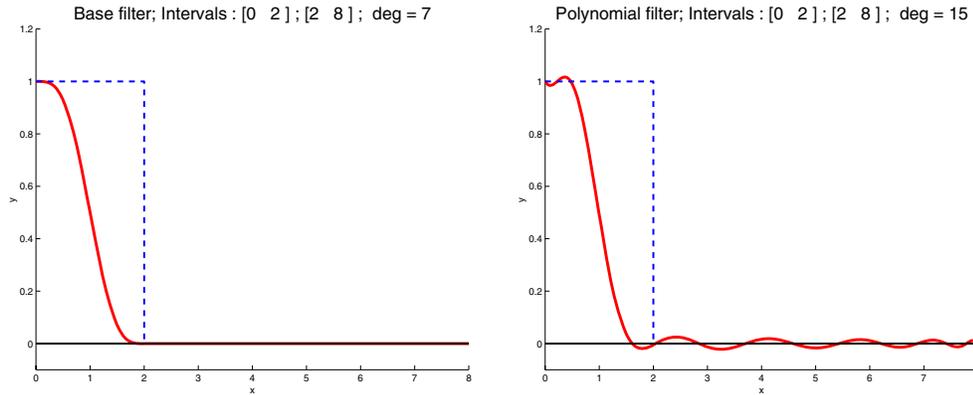
Base filter; Intervals : [0 2] ; [2 8] ; deg = 7

Polynomial filter; Intervals : [0 2] ; [2 8] ; deg = 15

FIG. 8. *The base filter $\Theta_{[4,4]}$ in $[0,2]$ and one in $[2,8]$ (left) and its polynomial approximation of degree* 15 *(right).*

Base filter; Intervals : [0 2] ; [2 8] ; deg = 11

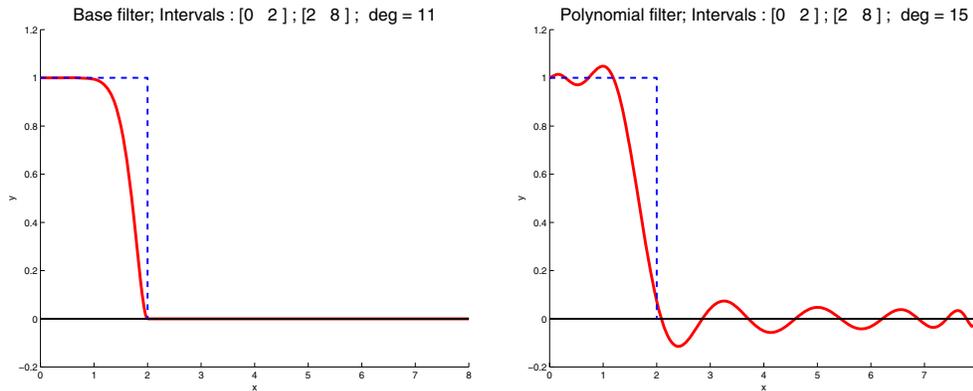Polynomial filter; Intervals : [0 2] ; [2 8] ; deg = 15

FIG. 9. *The base filter $\Theta_{[10,2]}$ in $[0,2]$ and one in $[2,8]$ (left) and its polynomial approximation of degree* 15 *(right).*

The bridge interval should be large enough so as to include all the eigenvalues of the wanted set, but not so large as to contain too many unwanted eigenvalues. In practice, augmenting the interval of wanted eigenvalues slightly only minimally hampers performance but helps improve the robustness of the procedure.

There are a number of parameters which can be exploited to yield a desired filter polynomial. 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 subinterval. Our experience shows that using an equal weighting scheme for each subinterval is a very reasonable choice for most applications, including electronic structure calculations.

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 other good reference points for a useful comparison would be the partially reorthogonalized Lanczos (which was used in [1]) and the implicitly restarted Lanczos iteration as it is implemented in the popular package ARPACK [16, 33]. We compare these two algorithms with the filtered Lanczos (F. Lanczos) algorithm with partial reorthogonalization.

All the experiments which follow were performed on an SGI Origin 2000 system using a single R12000 processor at 300 MHz clock. The filtered Lanczos code is available from the authors upon request. `F.Lanczos` 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` [35]. The convergence tolerance was set to 10^{-10} for all methods. Notice that although in electronic structure calculations the convergence tolerance is typically taken 2–3 orders of magnitude larger, `F.Lanczos` (similar to `ARPACK` and Lanczos) can be used for other applications as well (this is why we include the test matrix from the Harwell–Boeing collection). In order to conduct a rather strict test we have chosen the above convergence tolerance. For `ARPACK` the maximum dimension of the Lanczos basis was always set equal to twice the number of requested eigenvalues. Thus, the number of implicit QR steps in `ARPACK` was equal to the number of the wanted eigenvalues. We point out that these settings are typically used in `ARPACK`.

For `F.Lanczos` the number of filtered Lanczos iterations for the initial vector was set to 200, while the degree of the inner CR polynomial was 8. In the latest (stabilized) version of the code we use 2 intervals for the base function: one for the wanted and another for the unwanted ones. The degrees m_1, m_2 for the smooth base function are set to $m_1 = 5$ and $m_2 = 15$. The number of Lanczos iterations for the determination of the bounding interval $[\alpha, \beta]$ for the spectrum was determined by a convergence tolerance of 10^{-6}. The above settings were the same for all test cases.

For partial reorthogonalization we used the default parameters defined in the `PLANSO` code. It is worth mentioning that the maximum loss of orthogonality allowed was set to the square root of the machine precision.

In implicitly restarted techniques, such as those 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 iteration and the unrestarted Lanczos iteration with partial reorthogonalization to compute all eigenvalues that are smaller than 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 cost. This was previously shown in [1], however, at the important expense of additional memory, relative to implicitly restarted techniques. The experiments that follow clearly 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 [2]. In addition, we have also used a test matrix, namely, the `Andrews` matrix, from the University of Florida sparse matrix collection,[3] so as to give an example of applicability of our method in other applications as well. Table 1 provides the characteristics of the test matrices. For the Hamiltonians the number of the requested eigenvalues

---

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

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

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

generally correspond to physical properties of the corresponding molecular system. For example, $Si_{10}H_{16}$ has 28 occupied states, while $Si_{34}H_{36}$ has 86, $Ge_{87}H_{76}$ has 212, and $Ge_{99}H_{100}$ 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 F. 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. Observe that our choice of matrices spans different degrees of sparsity in order to demonstrate the effect of the latter on the overall cost, since the F. Lanczos algorithm makes heavy use of matrix-vector products.

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

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 F. Lanczos algorithm over ARPACK is more pronounced for the sparser test matrices. Thus, although the number of matrix-vector products in F. Lanczos increases relative to ARPACK, a significant gain results from avoiding the updating of a large number of eigenvectors, which standard methods do at every step.

The use of partial reorthogonalization is indeed beneficial in both F. Lanczos and Partial Lanczos. However, the main advantage of F. Lanczos is the reduction of traffic in memory. For example, let us look at the case of $Si_{34}H_{36}$ and $n_o = 200$ (last row of subtable, Table 2). Observe that F. Lanczos uses 640 basis vectors, while Partial Lanczos has to move 3580 basis vectors from memory. For ARPACK we have 30 restarts, and at each restart the algorithm will "touch" 400 vectors (twice the number of sought eigenvalues); thus we have a total of at least 12000 basis vectors moving between memory and CPU (including other costs such as full reorthogonalization).

In electronic structure calculations the required accuracy is close to $0.5 \cdot 10^{-6}$, which is larger than the semiorthogonality level of 10^{-8} that is ensured by partial reorthogonalization. However, in applications that have stricter accuracy requirements, semiorthogonality of basis vectors may not be adequate. If this is the case, then we can use full reorthogonalization in F. Lanczos. Of course, we can expect the benefits over ARPACK to reduce somewhat; however, the major improvement which results from the small basis of F. Lanczos and its unrestarted nature, and thus its much lesser use of memory, is still there. On the other hand, using a large convergence tolerance could prove tricky in ARPACK as this code relies on deflation techniques in order to improve

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 F.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.

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

convergence. Thus, poorly converged (large) eigenvectors will reenter in the iteration, slowing convergence towards small eigenvalues.

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 and a significant increase in memory traffic, and to dramatically prolong the run-times.

6. Conclusions. This paper presented a filtered Lanczos iteration 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, filtered Lanczos 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 reorthogonalization scheme [31]. One technique which filtered Lanczos 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 filtered Lanczos procedure applies a fixed, precomputed, 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 expediting convergence towards the desired invariant subspace. Earlier work (see, e.g., [28]) showed how one can design a CR-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 the filtered Lanczos algorithm.

Experimental evidence clearly shows that the new method 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.

REFERENCES

[1] C. BEKAS, Y. SAAD, M. TIAGO, AND J. CHELIKOWSKY, *Computing charge densities with partially reorthogonalized Lanczos*, Comput. Phys. Comm., 171 (2005), pp. 175–186.

[2] J. R. CHELIKOWSKY, L. KRONIK, I. VASILIEV, M. JAIN, AND Y. SAAD, *Using real space pseudopotentials for electronic structure calculations*, in Handbook of Numerical Analysis, Vol. 10, Special Volume on Computational Chemistry, C. Le Bris, ed., Elsevier Science B.V., Amsterdam, 2003, pp. 613–637.

[3] J. CULLUM AND R. A. WILLOUGHBY, *Lánczos Algorithms for Large Symmetric Eigenvalue Computations*, Vol. 1, Birkhäuser, Boston, 1985.

[4] J. CULLUM AND R. A. WILLOUGHBY, *Lánczos Algorithms for Large Symmetric Eigenvalue Computations*, Vol. 2, Birkhäuser, Boston, 1985.

[5] A. EDELMAN, T. A. ARIAS, AND S. T. SMITH, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 303–353.

[6] 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.

[7] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[8] P. HOHENBERG AND W. KOHN, *Inhomogeneous electron gas*, Phys. Rev., 136 (1965), pp. B864–B871.

[9] Y. HUANG, D. J. KOURI, AND D. K. HOFFMAN, *Direct approach to density functional the-*

ory: Iterative treatment using a polynomial representation of the heaviside step function operator, Chem. Phys. Lett., 243 (1995), pp. 367–377.

[10] L. O. JAY, H. KIM, Y. SAAD, AND J. R. CHELIKOWSKY, *Electronic structure calculations using plane wave codes without diagonalization*, Comput. Phys. Comm., 118 (1999), pp. 21–30.

[11] W. KOHN AND L. J. SHAM, *Self-consistent equations including exchange and correlation effects*, Phys. Rev., 140 (1965), pp. A1133–A1138.

[12] E. KOKIOPOULOU AND Y. SAAD, *Polynomial filtering in latent semantic indexing for information retrieval*, in Proceedings of the ACM-SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, 2004, pp. 104–111.

[13] G. KRESSE AND J. J. FURTHMÜLLER, *Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set*, Phys. Rev. B, 54 (1996), pp. 11169–11186.

[14] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Research Nat. Bur. Standards, 45 (1950), pp. 255–282.

[15] R. M. LARSEN, *Efficient Algorithms for Helioseismic Inversion*, Ph.D. thesis, Department of Computer Science, University of Aarhus, Aarhus, Denmark, 1998.

[16] R. B. LEHOUCQ, D. C. SORENSEN, AND C. YANG, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998, http://www.caam.rice.edu/software/ARPACK.

[17] W. Z. LIANG, C. SARAVANAN, Y. SHAO, A. BELL, AND H. HEAD-GORDON, *Improved Fermi operator expansion methods for fast electronic structure calculations*, J. Chem. Phys., 119 (2003), pp. 4117–4125.

[18] F. MAURI AND G. GALLI, *Electronic-structure calculations and molecular-dynamics simulation with linear system size scaling*, Phys. Rev. B, 50 (1994), pp. 4316–4326.

[19] F. MAURI, G. GALLI, AND R. CAR, *Orbital formulation for electronic-structure calculations with linear system size scaling*, Phys. Rev. B, 47 (1993), pp. 9973–9976.

[20] A. NEUMAIER, *Solving ill-conditioned and singular linear systems: A tutorial on regularization*, SIAM Rev., 40 (1998), pp. 636–666.

[21] A. M. N. NIKLASSON AND M. CHALLACOMBE, *Density matrix perturbation theory*, Phys. Rev. Lett., 92 (2004), article 193001.

[22] A. M. N. NIKLASSON, C. J. TYMCZAK, AND M. CHALLACOMBE, *Trace resetting density matrix purification in O(n) self-consistent-field theory*, J. Chem. Phys., 118 (2003), pp. 8611–8620.

[23] C. C. PAIGE, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, Ph.D. thesis, London University, London, 1971.

[24] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 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. Modern Phys., 64 (1992), pp. 1045–1097.

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

[27] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[28] Y. SAAD, *Filtered conjugate residual-type algorithms with applications*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 845–870.

[29] Y. SAAD, A. STATHOPOULOS, J. CHELIKOWSKY, K. WU, AND S. ÖĞÜT, *Solution of large eigenvalue problems in electronic structure calculations*, BIT, 36 (1996), pp. 563–578.

[30] A. H. SAMEH AND J. A. WISNIEWSKI, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal., 19 (1982), pp. 1243–1259.

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

[32] H. D. SIMON, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–142.

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

[34] G. W. STEWART, *Adjusting the Rayleigh quotient in semiorthogonal Lanczos methods*, SIAM J. Sci. Comput., 24 (2002), pp. 201–207.

[35] K. WU AND H. SIMON, *A Parallel Lanczos Method for Symmetric Generalized Eigenvalue Problems*, Report 41284, Lawrence Berkeley National Laboratory, 1997.

[36] C. YANG, J. MEZA, AND L. W. WANG, *A constrained optimization algorithm for total energy minimization in electronic structure calculation*, J. Comput. Phys., 217 (2006), pp. 709–721.