Diagonalization methods in PARSEC [*]

Yousef Saad [†] Yunkai Zhou [†] Constantine Bekas [‡]
Murilo L. Tiago [§] James R. Chelikowsky [§]

June 1st, 2006

Abstract

The Pseudopotential Algorithm for Real-Space Electronic Calculations (PARSEC) is a software package written by a team consisting of materials scientists and computer scientists over approximately one decade. During this period, the algorithms used in PARSEC have undergone a series of improvements. This paper reviews the nature of the eigenvalue problems encountered in solving the Kohn-Sham equations and discusses the evolution of the diagonalization methods used in PARSEC.

Keywords: Diagonalization, Kohn-Sham equations, Density Functional Theory, Lanczos algorithms, subspace iteration

1 Introduction: The Kohn-Sham Equations

The electronic structure of a condensed matter system, e.g., cluster, liquid or solid, is described by a wave function Ψ which can be obtained by solving the Schrödinger equation:

$$\mathcal{H}\Psi = \mathcal{E}\Psi, \tag{1}$$

where $\mathcal{H}$ is the Hamiltonian operator for the system and $\mathcal{E}$ the total energy.

In its original form the operator $\mathcal{H}$ is very complex, involving sums over all electrons and nuclei, Laplacian related to each nucleus, etc. However, most theories of condensed matter systems make two fundamental approximations which render the problem more tractable. These are the Born-Oppenheimer approximation and the one-electron approximation [1].

[*]Work supported by NSF under grants DMR/0325218 and DMR/0551195 and by the Minnesota Supercomputer Institute

[†]Computer Science & Engineering, University of Minnesota, 200 Union st. SE, Minneapolis, MN 55455. E-mail: {`saad,yunkai`}`@cs.umn.edu`

[‡]IBM Research, Zurich Research Laboratory, Saumerstrasse 4, 8003 Rüschlikon - Switzerland Email: bek@zurich.ibm.com

[§]Institute for Computational Engineering and Sciences (ICES) (C0200), ACES Building, Room 4.324, 201 East 24th Street ACES, 1 University Station, University of Texas at Austin, Austin, Texas 78712. Email: {`mtiago, jrc`} `@ices.utexas.edu.`

With these approximations the following simplified form of the Schrödinger equation, known as the Kohn-Sham equation, is obtained:

$$\left[\frac{-\hbar^2\nabla^2}{2m} + V_{tot}[\rho(r), r]\right]\psi(r) = E\psi(r), \tag{2}$$

Here, the Laplacian represents the kinetic operator, $\hbar$ is Planck's constant, m is the electron mass, and V_{tot} is the total potential at some point r in space. As the equation indicates, the potential depends on the charge density ρ whose expression will be given below. The pototential V_{tot} is the sum of three components: The ionic potential which reflects energy from the core electrons, the Hatree potential which reflects electron-electron Coulombic energies, and the Exchange-Correlation potential which arises from the one-electron approximation:

$$V_{tot} = V_{ion} + V_H + V_{xc} \ .$$

Both of the terms V_{xc} and V_H depend on the charge density $\rho(r)$, in a certain point in space. This charge density in turn depends on the wavefunctions of the above equation via,

$$\rho(r) = \sum_{\text{occupied states}} |\psi_i(r)|^2. \tag{3}$$

The numerical problems encountered in solving the above Kohn-Sham equations are among the most challenging in computational sciences today. The central computation involved is the repeated solution of a large, symmetric eigenvalue problem. Traditional approaches use a plane-wave basis to expand the required wave-functions (eigenvectors). This approach is comparable to spectral techniques used in solving other types of partial differential equations. In the mid-1990s, our group started exploring an alternative to this approach which is based on exploiting high-order finite difference schemes. For localized systems, the methods proved to be as accurate and more efficient than plane-wave techniques [2, 3, 4]. The matrices resulting from both finite difference methods and plane-wave techniques are large, and the number of eigenvalues and eigenvectors required is proportional to the number of atoms in the system. Herein lies the challenge in DFT calculations.

2 Solution methods

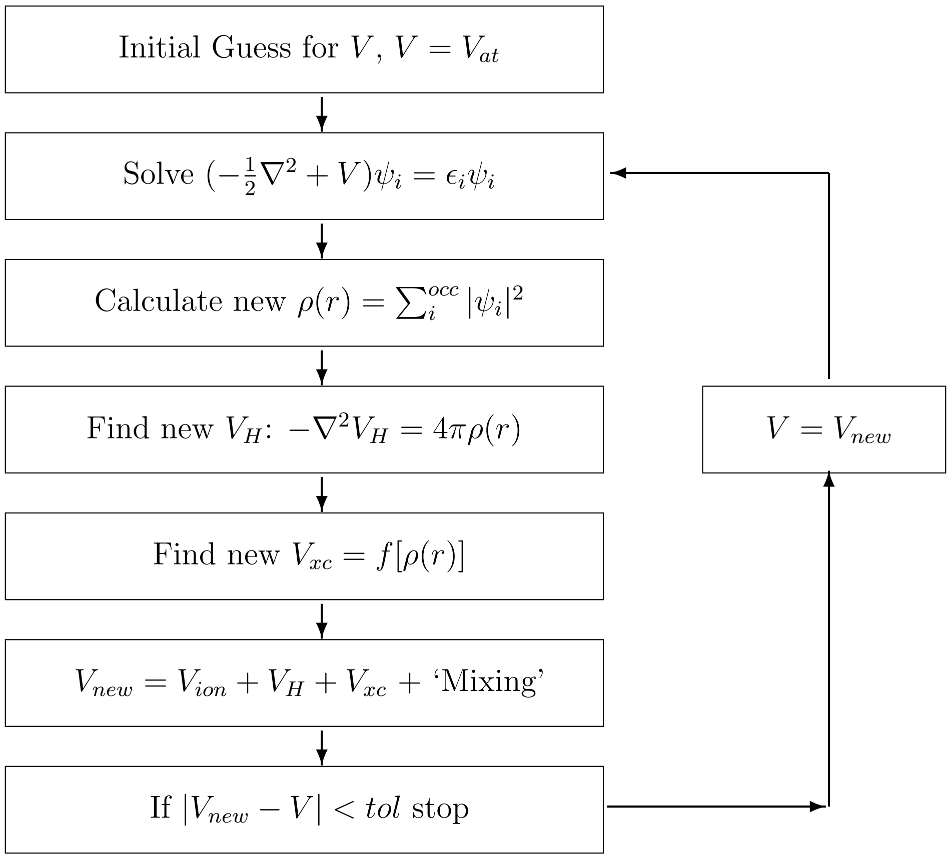
It is important to understand the nature of the problem embodied by the Kohn-Sham equations. The potential and charge density must be self-consistent in the Kohn-Sham equations. This means that the charge density (for example) as defined by (3), where the wavefunctions ψ_i are solutions of (2), should be identical with the charge density used in solving (2). Therefore, one can view the KS equations under different angles:

1. A nonlinear eigenvalue problem ;

2. A system of nonlinear equations;

3. A nonlinear optimization problem whereby the charge density and wavefunctions are sought to minimize total energy.

In practice, the problem is viewed as a nonlinear eigenvalue problem, where the nonlinearity is treated by an SCF (self-consistend field) iteration which exploits a Broyden-type quasi-Newton approach. In most situations the nonlinear SCF iteration takes a few steps, though the iteration may encounter convergence difficulties for metallic systems.



Initial Guess for V, $V = V_{at}$
Solve $(-\frac{1}{2}\nabla^2 + V)\psi_i = \epsilon_i\psi_i$
Calculate new $\rho(r) = \sum_i^{occ} |\psi_i|^2$
Find new V_H: $-\nabla^2 V_H = 4\pi\rho(r)$
Find new $V_{xc} = f[\rho(r)]$
$V_{new} = V_{ion} + V_H + V_{xc}$ + 'Mixing'
If $|V_{new} - V| < tol$ stop
$V = V_{new}$

Figure 1: Schematic representation of the SCF loop

A schematic representation of the SCF loop is shown in Figure 1. The most time consuming part of the calculation is in solving the eigenvalue problems. The main difficulty is not just the sheer size of the problem but rather the fact that the number of wanted eigenvalues (occupied states) can be quite large. It has often been observed, though not emphasized in the scientific computing community, that in this situation, the orthogonalization process often constitutes a significant part of the cost of the whole diagonalization process. It is therefore important to try to consider methods which attempt to reduce the orthogonalization cost. Several of the methods considered recently have been motivated by this observation.

3 Real-space Finite Difference Methods and PARSEC

Efforts by our team to develop an effective real-space package began over a decade ago. These resulted in the recently released Pseudopotential Algorithm for Real Space Electronic

Calculations (PARSEC) code. This software represents the collaborative efforts of several researchers (see a partial list in the acknowledgments at the end of the paper). It runs on parallel as well as sequential platforms. The parallel option runs as a distributed memory program, using MPI for communication. Many improvements, some major, others minor, were added over the years. The last two major changes are (1) procedure to take advantage of (geometric) symmetry (2) inclusion of new diagonalization methods. The following illustrates some milestones of the code development:

- Sequential real-space code developed on a Cray YMP [up to '93]
- Ported to a cluster of SGI workstations [up to '96]
- CM5 ['94-'96] (Massive parallelism begins).
- IBM SP2 (Using PVM)
- Cray T3D (Combining PVM + MPI) – around '96-'97
- Cray T3E (using MPI) – '97
- IBM SP with +256 nodes – '98+
- SGI Origin 3900 [128 processors] – '99+
- IBM SP - PARSEC name given, '02
- SGI Altix, IBM SP4, and other platforms. Code rewritten in F90. PARSEC V.1.0 released in '05.

PARSEC discretizes the Kohn-Sham equations in real space by using High-Order Finite Difference Methods. The typical base geometry is a cube from which points outside of a sphere are removed. The advantages of using finite differences on regular meshes are many and have been emphasized elsewhere [5, 2, 3, 4]. For example, the matrix associated with the Laplacian need not be actually stored. Indeed, the result of multiplying it by a vector can be implemented using "stencils" [6]. Second, implementation aspects of the code are simplified, relative to planewave-based codes. Thus, parallelization, based on a domain-decomposition approach, becomes straightforward.

In recent years, electronic structure codes based on real space finite difference implementations have become increasingly popular and gained ground over the standard planewave-based codes. They now allow to handle very large systems effectively and accurately on massively parallel computers.

One question which is often asked is: Why not use Finite Element Method (FEM) discretizations? Finite elements allow to reduce the number of discretization points by refining the mesh only where this is necessary. However, using an FEM approach with irregular meshes, would be far more complex to implement. In particular, any benefits gained from using FEM meshes, namely the reduction of the number of points required, may be outweighed by the loss of simplicity and the additional cost of indirect addressing inherent to irregular sparse computations. Using high-order finite differences allows to reduce the number of points required – possibly in a better way – without having to sacrifice simplicity.

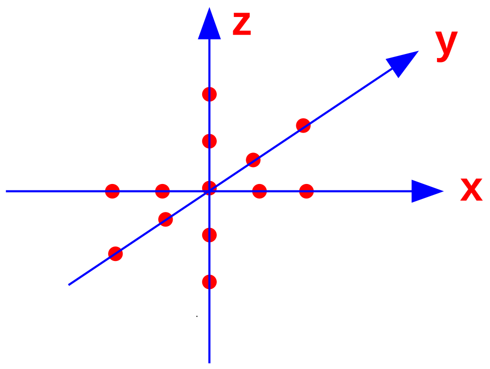


Figure 2: Order 4 Finite Difference Approximation stencil.

Our current code uses up to order-12 finite difference formulas, but typically order six to eight formulas are more common. A stencil of order 4 is shown if Figure 2. Using higher degree formulas in order to reduce the number of points further does not pay in general because of other limitations and inaccuracies in other parts of the method. An example of a nonzero pattern of a sparse matrix resulting from the FD discretization is shown in Figure 3. Recall however, that even though the matrix is sparse, it is not stored as a sparse matrix.

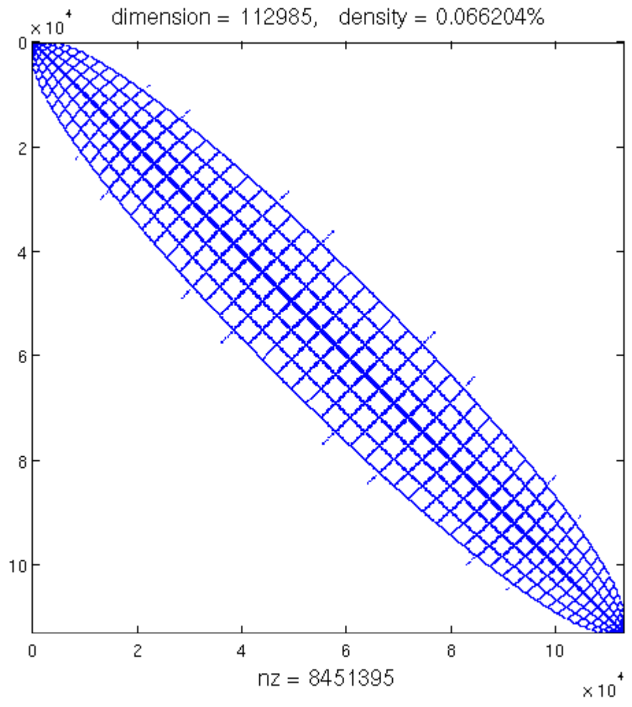


Figure 3: Pattern of resulting matrix for Ge99H100

As an example of the improvements made by the code over the past few years, consider the example of the quantum dot reported in [5]. One particular system discussed in the paper is an $Si_{525}H_{276}$ cluster. This particular system leads to a discretized Hamiltonian of size $N \approx 290,000$ and a number of states $n_{states} = 1,194$. The experiment performed in around 1997, took about 20 hours of CPU time to complete on a Cray T3E, using 48 processors. The exact same example calculated with the latest version of PARSEC, takes only two hours, *on one Processor* [7]. The processor is an Intel Madison processor with a clock rate of 1.3GHz. This means that the same calculation which could only be done with sizable super-computer resources in 1997, can now essentially be handled by a good workstation. Needless to say, as is always the case, the gains are to be attributed to improvements in both hardware and algorithms. On the algorithms side, the two main ingredients to the gains are (1) more efficient diagonalization (2) inclusion of capability to exploit symmetry. The next section explores the diagonalization methods used.

4 Diagonalization methods in PARSEC

Diagonalization methods used in the early versions of PARSEC emphasized simplicity and robustness. We used an algorithm, called DIAGLA, based on a block-Davidson approach with various enhancements. One of the important features of DIAGLA is its ability to use parts of subspaces from previous SCF iterations as a starting point to a new SCF loop. This was the work-horse for the earlier versions of PARSEC until about 2000 when we explored the usefulness ARPACK [8], a well-tuned public domain package which became available a few years earlier. ARPACK was added as an option, as it was found to be competitive in general, and superior in many cases to DIAGLA.

An inconvenient feature of ARPACK is that it does not have the ability to reuse previous spaces, as did DIAGLA, for starting new SCF loops. It was also not easily amenable to changes that would enable this feature. In fact, ARPACK can be viewed as a highly effective method for computing a relatively small number of eigenvectors, but it was not designed for computing large eigenspaces.

4.1 Focus on eigenspaces

In our recent work this observation was taken a step further and we considered methods that are specifically designed for computing large eigenspaces instead of focussing on eigenvalues. The rationale for this viewpoint is that most eigenvalue codes over-emphasize the accuracy of individual eigenvectors at substantial cost. In particular, as in Lanczos-type methods, large subspaces are needed in order to extract accurate eigenvectors. This implies higher orthogonalization costs if full reorthogonalization is used.

4.2 Partial Reorthogonalization Lanczos

A first approach we attempted for computing invariant subspaces was a Partial Reorthogonalization Lanczos algorithm [9, 10].

We set as a goal to primarily consider the problem as an eigenspace problem instead of an eigenvalue problem. The difference between these two viewpoints is that an eigenspace may have a basis other than an eigenbasis, and this basis may be accurate enough for computing charge densities.

The charge density $\rho(r)$ at a point r in space is traditionally computed from the eigenvectors Ψ_i of the Hamiltonian matrix H via the formula

$$\rho(r) = \sum_{i=1}^{n_o} |\Psi_i(r)|^2, \tag{4}$$

where the summation is taken over all occupied states of the system under study. However, it is also possible to compute $\rho(r)$ without explicitly resorting to using eigenvectors. Let the vectors ψ_i be the discretizations of $\Psi_i(r)$ with respect to r. Then, the charge densities are the diagonal entries of the "functional density matrix"

$$P = V_{n_o} V_{n_o}^\top \quad \text{with} \quad V_{n_o} = [\psi_1, \ldots, \psi_{n_o}]. \tag{5}$$

Specifically, the charge density at the j-th point r_j is the j diagonal entry of P. An important observation here is that any orthogonal basis $\mathcal{V}$ which spans the same subspace as the eigenvectors $\psi_i,\ i = 1, \ldots, n_o$ can be used, not just the eigenbasis.

In [11] a Lanczos procedure was used to generate a good subspace from which an apropriate basis is extracted to use in Equation (5). The algorithm is based on the Lanczos procedure with partial reorthogonalization and the use of a special stopping criterion for determining when the underlying desired subspace has copnverged. The Lanczos procedure is used without restarts and this results in larger bases than would normally be required by a standard restarted algorithm.

Recall the Lanczos recurrence:

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$$

Here the scalars β_{j+1}, α_j are selected so that $v_{j+1} \perp v_j$, $v_{j+1} \perp v_{j-1}$, and $\|v_{j+1}\|_2 = 1$. In theory this is enough to guarantee that the whole system $\{v_j\}$ is orthonormal and using the notation $V_m \equiv [v_1, \ldots, v_m]$, we have:

$$V_m^T A V_m = T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{bmatrix}$$

However, in practice the algorithm undergoes a severe loss of orthogonality as soon as first eigenvalues start to converge. The remedy is to use some form of reorthogonalization. In *Partial Reorthogonalization* the current Lanczos vector is reorthogonalized against all previous vectors only when this is deemed necessary. This can be done thanks to an inexpensive recurrence relation [9, 12, 13, 14, 15]. In recent work [14, 15] this idea which dates from the mid-80s was revived and a code became available. Our tests with real-space Hamiltonians from PARSEC, revealed that the need for reorthogonalization is not too strong, see Figure 4 for an example.

The second important ingredient of this procedure is a test whose goal is to indicate how good is the underlying eigenspace *without knowledge of individual eigenvectors.* When the test indicates that the underlying subspace has converged – then we can compute the desired

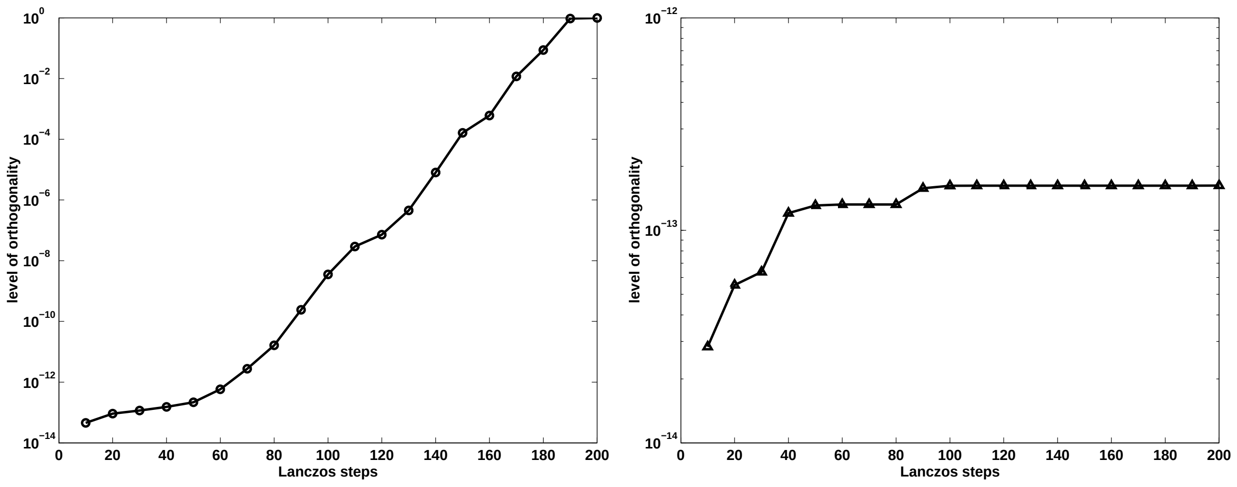


Figure 4: Loss of orthogonality in standard Lanczos (left) and recovery of orthogonality with partial reorthogonalization Lanczos (right). The curves show the levels of orthogonality of the Lanczos basis in both cases, for a Hamiltonian of size $n = 17077$ corresponding to $\mathtt{Si_{10}H_{16}}$. Note that a total of 34 reorthogonalizations were taken.

	Partial Lanczos				ARPACK			
n_o	$A * x$	orth	mem.	secs	$A * x$	rest.	mem.	secs
248	3150	109	2268	2746	3342	20	357	16454
350	4570	184	3289	5982	5283	24	504	37371
496	6550	302	4715	13714	6836	22	714	67020

Table 1: Partial Reorth. Lanczos vs. ARPACK for $Ge_{99}H_{100}$.

orthogonal basis (eigenvectors). This test is inspired from the meaning of the sum of eigenvalues associated with the occupied states, which correspond to energies. The underlying eigenspace is deemed to have converged when this total energy has converged. This requires the computation of the sum of the n_o lowest eigenvalues of the tridiagonal matrix generated by the Lanczos algorithm, where n_o is the number of occupied states.

Table 1 shows an example of the performance of this approach, see also, [11]. Here the matrix size is $N = 94,341$; and the number of nonzeros is $nnz = 6,332,795$. The number of occupied states is 248. Generally, the approach requires more memory than ARPACK or other standard diagonalization methods, but this results in a factor of 4 to 6 gain in CPU time. On the memory issue, one may note that in most cases we have tested, the demand was still manageable. In addition, there is the option of using secondary storage which, if properly implemented, should bypass problems with storage. The above mentioned partial Lanczos is not implemented in PARSEC. On the other hand, we have integrated the Thick-Restart Lanczos (TRLan) code [16, 17] into PARSEC [7] because it requires less memory.

4.3 Chebyshev Subspace Iteration

A further improvement of the idea of exploiting subspaces, can be obtained by resorting to a subspace iteration algorithm which exploits Chebyshev filtering. The main idea of Chebyshev subspace iteration is not new, see for example, [9, 18, 19]. In fact the classical subspace iteration has long been viewed as incompetitive relative to the Lanczos algorithm. An algorithm which is sub-optimal in a general context, can become the method of choice in a specific context. This has to do with the way in which the algorithm is used and with some appealing features of subspace iteration. In particular, subspace iteration is easier to extend to a nonlinear version.

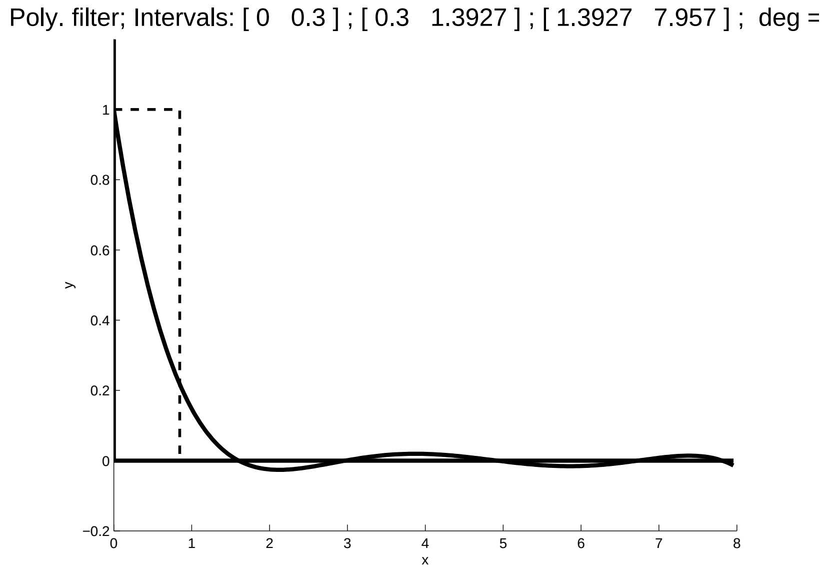


Figure 5: A typical Chebyshev filter.

Given a subspace $V_i = [v_1, \ldots, v_m]$, the basic step of the algorithm is to "filter" the subspace as

$$\hat{V}_{i+1} = P_k(A)V_i$$

where P_k is a polynomial of low degree selected to enhance desired eigencomponents of each of the vectors v_i. Instead of iterating the subspace to convergence as is done in the traditional subspace iteration, the procedure we propose avoids this and accepts whatever improvements have been made by filtering, to directly proceed with the next SCF iteration. In other words *diagonalization has been replaced by subspace filtering.* Another way to view this is that the SCF loop and the diagonization loop are merged into a form of "nonlinear subspace iteration". A flowchart of the algorithm is shown in Figure 6. A question is whether this will have a negative effect on convergence. We observed that, in most cases, the method converges within similar number of iterations as eigenvector-based methods do.

Extensive numerical experiments available in [7, 20] indicate that the new algorithm is several times faster than methods based on standard iterative diagonalizations. We reproduce here a typical example with the silicon cluster $Si_{525}H_{276}$ example mentioned earlier. For this example the matrix size is $292,584$ and the number of occupied states is 1,194. However, the system benefits from a 4-fold symmetry which leads to actual diagonalizations with four

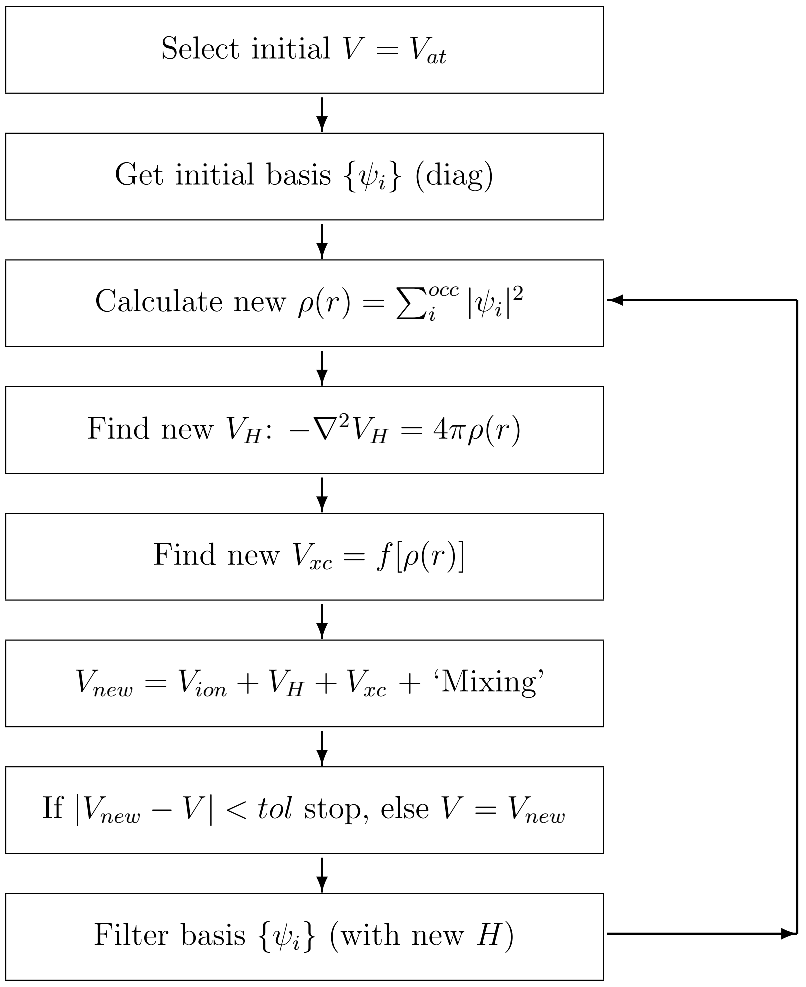

Select initial $V = V_{at}$
Get initial basis $\{\psi_i\}$ (diag)
Calculate new $\rho(r) = \sum_i^{occ} |\psi_i|^2$
Find new V_H: $-\nabla^2 V_H = 4\pi\rho(r)$
Find new $V_{xc} = f[\rho(r)]$
$V_{new} = V_{ion} + V_H + V_{xc} +$ 'Mixing'
If $|V_{new} - V| < tol$ stop, else $V = V_{new}$
Filter basis $\{\psi_i\}$ (with new H)

Figure 6: Schematic representation of the SCF loop using Chebyshev filters instead of regular diagonalization

matrices of size $73,146$. The tests were performed on one 1.3 GHz Intel Madison processor of the SGI Altix cluster at the Minnesota Supercomputing Institute.

Table 2 shows a comparison of Chebyshev filtering (CheFSI) versus ARPACK and Thick-Restart Lanczos (TRLan, [16, 17]) within PARSEC. CheFSI uses degree-8 polynomials, with the first step diagonalization done by calling TRLan. All three methods exploit the same symmetry operations. We also note that the total energies calculated by the three methods agree to more than eight digits, more than the accuracy provided by DFT approximations.

5 Eigenvalue-free methods

As the size of the eigenvalue problem in the Kohn-Sham equation increases for larger systems, it is natural to ask how far one can go with classical approaches. It is important to remember that no matter what improvements are made to the algorithms, the scaling of an eigenvector-based approach is at least cubic with the number of states. For example, in the case of the

method	# $A * x$	SCF its.	CPU(secs)
CheFSI	124761	11	5946.69
ARPACK	142047	10	62026.37
TRLan	145909	10	26852.84

Table 2: ChebSI versus 2 standard diagonalization schemes for $Si_{525}H_{276}$

Chebyshev-Subspace iteration, the overall cost of the orthogonalization process scales like $n \times p^2$ where p is the number of states and n is the number of mesh-points. The number of mesh-points is indirectly related to the number of atoms, and it is ultimately at least linear with respect to this number. This is because it has to reflect the volume of the overall geometry. In addition, as p becomes larger, we should no longer ignore the burden of solving the projected problem, a dense eigenvalue problem whose cost is cubic with respect to size, i.e., with respect to the number of states. In fact, as the systems will become ever larger this cost will start exceedingly high. As an example, in a recent calculation done for a Silicon cluster of about 9,000 silicon atoms, the discretized Hamiltonian had a size close to 3 millions, and the number of states was close to 20,000.

Therefore, it is important to ask whether or not we can completely avoid diagonolization and replace it by a much more economical procedure. There exist techniques which avoid solving eigenvalue problems (or equivalent optimization problems). The main ingredient of these techniques, to which the class of "Order(n) methods" or "linear scaling methods" belong, is to obtain the charge density by other means than using eigenvectors, e.g. by exploiting certain decay features of the density matrix $\rho(r, r')$.

Recall that the density matrix is defined as $P = f(H)$, where f is a step function. We can approximate f by, e.g., a polynomial and this will result in a procedure to obtain columns of P inexpensively via: $Pe_j \approx p_k(H)e_j$. Doing this for $j = 1 : n$ will not result in a practical procedure. However, one can exploit the sparsity of P (especially in planewave bases). Here, ideas of "probing" allow to compute several columns of P at once. The probing technique is illustrated in Figure 7. In a nutshell, when (and if) P is sparse and its sparsity pattern is known, it is possible to compute all the columns that do not share common nonzero entries at once with a single multiplication with a well selected vector. For example, for a tridiagonal matrix, columns $1, 4, 7, 10, (3k-2), ...$ can all be computed at once by multiplying it with the vector $[1, 0, 0, 1, 0, 0, 1, 0, 0, ..]^T$. Similarly, it is possible to compute colums $3k - 1$ for $k = 1, \ldots, .., \lceil n/3 \rceil$ with a single matrix-vector product, and then colums $3k$ for $k = 1, \ldots, .., \lceil n/3 \rceil$ with a single matrix-vector product. In all the entire matrix can be obtained in 3 matvecs. This can be extended to general sparse matrices provided the pattern is known in advance.

Probing computes the whole matrix P whereas we are interested only in its diagonal which represents the charge density. One can ask the question as to what can be done if the pattern is known and we seek only the diagonal. This was explored for the case of banded matrices, in [21].

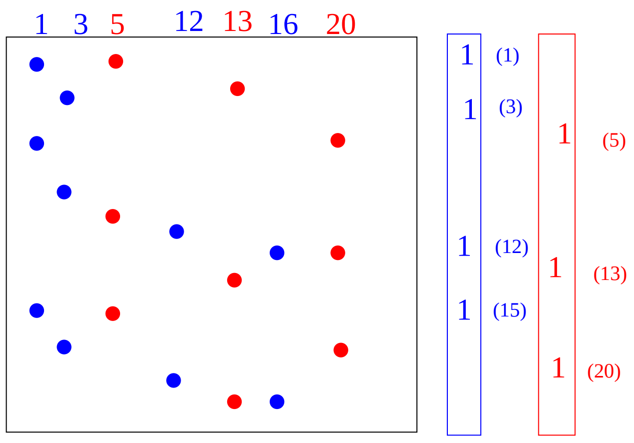


Figure 7: Probing in action: blue columns can be computed at once by one matrix-vector product. Then red columns can be coumputed the same way.

6 Conclusion

In exploring various techniques for solving the eigenvalue problems in DFT, we observed that to improve efficiency one must exploit two important factors: (1) the problem is an eigenspace problem instead of a standard eigenvalue problem; (2) it is important to take into account the outer nonlinear loop. Observation (1), means that we need to pay less attention to the convergence of the individual eigenvectors. Instead, a subspace can be built which has just the required size, and can be improved by operating with the Hamiltonian. The result is a dramatic reduction in diagonalization costs. The second observation enables us to merge the SCF and diagonalization loops and consider the problem as fully nonlinear.

It remains to consider what can be done for much larger systems. Existing "linear scaling methods" may be adapted by adding techniques from linear algebra, such as probing, or some of the techniques seen in [21].

Acknowledgements

The PARSEC work is an on-going team effort. Among the main contributors to the code are (in an approximatly chronological order): J. Chelikowsky, Y. Saad, N. Troullier, K. Wu, X. Jing, H. Kim, A. Stathopoulos, I. Vassiliev, M. Jain, L. Kronik, A. Makmal, R. Burdick, M. Alemany, M. Tiago, Y. Zhou. Many others have contributed in other ways, with algorithmic ideas or with technical support.

References

[1] W. Kohn and L. J. Sham. *Phys. Rev.*, 140:1133–, 1995.

[2] J. R. Chelikowsky, N. Troullier, and Y. Saad. The finite-difference-pseudopotential method: Electronic structure calculations without a basis. *Physical Review Letters*, 72:1240, 1994.

[3] J. R. Chelikowsky, N. Troullier, K. Wu, and Y. Saad. Higher order finite difference pseudopotential method: An application to diatomic molecules. *Physical Review B*, B 50:11355, 1994.

[4] X. Jing, N. Troullier, D. Dean, N. Binggeli, J. R. Chelikowsky, K. Wu, and Y. Saad. Ab initio molecular dynamics simulations of Si clusters using a high-order finite-difference-pseudopotential method. *Physical Review B*, B 50:12234, 1994.

[5] A. Stathopoulos, S. Öğüt, Y. Saad, J.R. Chelikowsky, and H. Kim. Parallel methods and tools for predicting materials properties. *Computing in Science and Engineering*, 2:9–18, 2000.

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

[7] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky. Self-consistent-field calculation using Chebyshev polynomial filtered subspace iteration. *Journal of Computational Physics*, (to appear).

[8] 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`.

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

[10] 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.

[11] C. Bekas, Y. Saad, M. L. Tiago, and J. R. Chelikowsky. Computing charge densities with partially reorthogonalized Lanczos. *Computer Physics Communications*, 171(3):175–186, 2005.

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

[13] H. D. Simon. The lanczos algorithm with partial reorthogonalization. *Math. Comp.*, 42:115–142, 1984.

[14] R. M. Larsen. PROPACK: A software package for the symmetric eigenvalue problem and singular value problems on Lanczos and Lanczos bidiagonalization with partial reorthogonalization, SCCM, Stanford University
URL: `http://sun.stanford.edu/∼rmunk/PROPACK/`.

[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] K. Wu, A. Canning, H. D. Simon, and L.-W. Wang. Thick-restart lanczos method for electronic structure calculations. *J. Comput. Phys.*, 154:156–173, 1999.

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

[18] H. Rutishauser. Computational aspects of F. L. Bauer's simultaneous iteration method. *Numerische Mathematik*, 13:4–13, 1969.

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

[20] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky. Accelerating self-consistent-field calculations using Chebyshev-filtered subspace iteration — the parallel version. Technical report, Minnesota Supercomputing Institute, University of Minnesota, 2006 (in preparation).

[21] C. Bekas, E. Kokiopoulou, and Y. Saad. An estimator for the diagonal of a matrix. *Applied Numerical Mathematics*, 2007. To appear.