Journal of Computational Physics 229 (2010) 453–460

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Iterative diagonalization in augmented plane wave based methods in electronic structure calculations

P. Blaha [a,*], H. Hofstätter [b], O. Koch [c], R. Laskowski [a], K. Schwarz [a]

[a] Institute of Materials Chemistry, Vienna University of Technology, Getreidemarkt 9/165-TC, A-1060 Vienna, Austria
[b] Institute of Mathematics, University of Vienna, Nordbergstrasse 8-10, A-1090 Vienna, Austria
[c] Institute for Analysis and Scientific Computing, Vienna University of Technology, Wiedner Hauptstrasse 8-10, A-1040 Vienna, Austria

ARTICLE INFO

Article history:
Received 23 October 2008
Received in revised form 12 August 2009
Accepted 24 September 2009
Available online 4 October 2009

MSC:
65F10
65F15
65Z05
81-08

Keywords:
Iterative diagonalization
Davidson method
Computational materials science
WIEN2k
APW
Augmented plane waves

ABSTRACT

Due to the increased computer power and advanced algorithms, quantum mechanical calculations based on Density Functional Theory are more and more widely used to solve real materials science problems. In this context large nonlinear generalized eigenvalue problems must be solved repeatedly to calculate the electronic ground state of a solid or molecule. Due to the nonlinear nature of this problem, an iterative solution of the eigenvalue problem can be more efficient provided it does not disturb the convergence of the self-consistent-field problem. The blocked Davidson method is one of the widely used and efficient schemes for that purpose, but its performance depends critically on the preconditioning, i.e. the procedure to improve the search space for an accurate solution. For more diagonally dominated problems, which appear typically for plane wave based pseudopotential calculations, the inverse of the diagonal of $(H - ES)$ is used. However, for the more efficient "augmented plane wave + local-orbitals" basis set this preconditioning is not sufficient due to large off-diagonal terms caused by the local orbitals. We propose a new preconditioner based on the inverse of $(H - \lambda S)$ and demonstrate its efficiency for real applications using both, a sequential and a parallel implementation of this algorithm into our WIEN2k code.

1. Introduction

In the last years quantum mechanical calculations based on density functional theory (DFT) [8] became very powerful due to the development of faster computers and more efficient algorithms. The key problem of such calculations is the solution of the Kohn–Sham (KS) equation [12]

$$\widehat{H}\Psi_i(r) = \left[-\frac{1}{2}\nabla^2 + v_{eff}(r)\right]\Psi_i(r) = \epsilon_i\Psi_i(r), \tag{1}$$

where the Hamilton operator $\widehat{H}$ contains the kinetic energy operator and an effective potential $v_{eff}(r)$. The latter depends on the electron density $\rho(r)$, which can be obtained from the wave functions Ψ_i, but obviously Ψ_i itself depends on $v_{eff}(r)$ and thus on $\rho(r)$. Hence, an iterative solution of the KS equations must be found by fixed point iteration, i.e. starting with some

* Corresponding author.
E-mail address: pblaha@theochem.tuwien.ac.at (P. Blaha).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.09.036

initial guess for $\rho(r)$ and iterating until the input and output densities, $\rho_{in}(r)$ and $\rho_{out}(r)$, becomes nearly the same (within a threshold). This is called the self-consistent-field (scf) procedure.

Each wave function Ψ is usually expanded in a set of (non-orthogonal) basis functions $\phi(r)$

$$\Psi(r) = \sum_j c_j \phi_j(r) \tag{2}$$

and the best wave functions are determined by a Rayleigh–Ritz procedure using the variational principle which leads to a generalized eigenvalue problem (see Section 2, Eq. (3)). The dimension of this problem depends on the investigated system (mainly on the number of atoms) and on the choice of the basis set ϕ, but typically varies for large scale problems from 10^3 up to 10^5. As mentioned above, the KS equations have to be solved iteratively, thus the corresponding eigenvalue problem does not need to be solved exactly at the beginning of the scf-iterations, but an approximate solution is fine as long as it does not affect the convergence of the scf-procedure. Note that the changes in the eigenvalue problem become smaller and smaller once the scf-procedure is nearly converged since v_{eff} will hardly change anymore. In addition, for most applications not all eigenvalues are needed, but for instance in the linearized augmented plane wave (LAPW), or its more efficient variant, the augmented plane wave plus local orbital (APW + lo) method implemented in the WIEN2k code [2,3,17,20,21], only the lowest 3–10% of all ϵ_i are needed.

The problem as sketched above calls for an iterative diagonalization scheme, since an accurate solution of the eigenvalue problem is only necessary at the very end of the scf-procedure and thus significant computer time can be saved in comparison to standard diagonalization procedures (this will be called "full diagonalization" later in this paper) based on LAPACK routines [9] for Cholesky factorization of the overlap matrix S (zpotrf), reduction to standard eigenvalue problem (zhegst), tri-diagonalization and solving for the lowest k eigenvalues and eigenvectors (zheevx) and finally backtransformation of the eigenvectors to the original generalized eigenvalue problem (ztrsm).

The iterative method for the diagonalization of generalized eigenproblems implemented so far in the WIEN2k code [3] is a blocked version of the Davidson method [4,5] which was introduced in [22]. Iterative methods for the problem at hand are also discussed in [1,13–15,18,25,28].

In [28], the method of *RMM–DIIS (residual minimization/direct inversion in the iterative subspace)* is proposed and compared with the *Davidson* and *Block Davidson* methods. The latter has the disadvantage that the doubling in the dimension of the search space is prohibitive for large initial subspaces. Therefore the RMM–DIIS method is claimed to have the advantage that only matrices of the size of the number of previous iteration steps are necessary. However, in its original version the method is fundamentally sequential in nature which the authors recognize as a major drawback [28], and which in the light of the development of parallel and grid-enabled versions of the WIEN2k code makes this approximate diagonalization unattractive. Recently, a reformulation of RMM–DIIS [19] has brought this method into the scope for a parallel implementation, however. The performance reported in [19] will be compared to our method later on. Another interesting approach was put forward in [27] where preconditioners similar to ours (based on approximations to the invers of $(H - \lambda S)$) were tested. However, these methods are designed for sparse matrices and insulating systems, while our method is applicable also to magnetic and metallic systems, where the occupation of the eigenvalues can significantly change during the scf-cycle.

A comparison with several other methods shows that (disregarding computational cost) the block Davidson method displays the best improvement in accuracy per iteration step due to the doubling of the search space [28]. Our aim is to avoid this doubling of the subspace.

Ref. [25] gives an overview of the state of the art of iterative diagonalization at that time, and demonstrates that a new *preconditioned conjugate gradient method* compares most favorably with *conjugate gradients*, *steepest descent* and *imaginary time propagation*. The VASP code [15], which is a highly efficient plane wave pseudopotential code, uses the RMM–DIIS method of [28] in a variant proposed in [18]. They claim this method to be superior for very large problems [13] if an unblocked, band-by-band iteration is used.

In this paper, we propose an approximate diagonalization which is motivated by the fact that the Davidson method previously implemented in the WIEN2k code [22] has proven to be no longer satisfactory once the basis set has been changed from the standard LAPW to the APW + lo basis set [17]. Apparently, the underlying discretization, the importance of non-diagonal terms (the local orbital contribution to the plane wave basis) and the adaptive basis set (the basis set changes slightly with v_{eff}) makes the preconditioning with only the diagonal elements $diag^{-1}[H - ES]$ not efficient anymore. Our new method is motivated by the improvements promised by the Jacobi–Davidson method [23,24,26] as compared to the original Davidson method [5]. However, application of the subspace expansion from the Jacobi–Davidson method seems prohibitively expensive, hence we propose a simplification which uses an approximate computation of the subspace expansion related to an iterative solution of the associated linear system of equations. Furthermore, we are going to demonstrate in Section 4 that the method is superior to full diagonalization in both efficiency and parallelism.

2. New diagonalization method

We want to compute the k eigenpairs corresponding to the lowest eigenvalues of the generalized eigenproblem

$$HY = SY\Lambda, \tag{3}$$

where $H \in \mathbb{C}^{n\times n}$ is hermitian and $S \in \mathbb{C}^{n\times n}$ is hermitian and positive definite.[1] The solution of (3) consists of the (matrix of the) lowest eigenvalues $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$ and matrix $Y \in \mathbb{C}^{n\times k}$ containing k corresponding eigenvectors y_j. The algorithm proceeds as follows:

- Input:

$$Y = [y_1, \ldots, y_k] \in \mathbb{C}^{n\times k}. \tag{4}$$

  Usually these are approximations to eigenvectors which were computed in the last scf-cycle.
- Compute the Ritz values (Rayleigh quotients)

$$\vartheta_j = \frac{y_j^\dagger H y_j}{y_j^\dagger S y_j}, \quad j = 1, \ldots, k. \tag{5}$$

- Set up the search space $[Y\ Z] \in \mathbb{C}^{n\times 2k}$ with

$$z_j = (H - \bar{\lambda}S)^{-1}(H - \vartheta_j S)y_j, \quad j = 1, \ldots, k. \tag{6a}$$

  or

$$z_j = (H_0 - \bar{\lambda}S_0)^{-1}(H - \vartheta_j S)y_j, \quad j = 1, \ldots, k. \tag{6b}$$

  where H_0 and S_0 are the matrices H and S of some previous (usually the first) scf-cycle, so the inverse $(H_0 - \bar{\lambda}S_0)^{-1}$ does not have to be recalculated in each scf-cycle. $\bar{\lambda}$ denotes some parameter for the tuning of the algorithm, see the remarks below.
- Set up the reduced problem

$$\tilde{H}V = \tilde{S}V\Gamma, \tag{7}$$

  where

$$\tilde{H} = [Y\ Z]^\dagger H [Y\ Z] = \begin{bmatrix} Y^\dagger HY & Y^\dagger HZ \\ Z^\dagger HY & Z^\dagger HZ \end{bmatrix} \tag{8}$$

  and

$$\tilde{S} = [Y\ Z]^\dagger S [Y\ Z] = \begin{bmatrix} Y^\dagger SY & Y^\dagger SZ \\ Z^\dagger SY & Z^\dagger SZ \end{bmatrix}. \tag{9}$$

- Compute eigenvectors $V_{1:k}$ of (7) corresponding to the lowest k eigenvalues $\gamma_1 \leqslant \cdots \leqslant \gamma_k$. Here and in the following, $V_{1:k}$ refers to the first k columns of a matrix $V \in \mathbb{C}^{n\times m}$ for any $m \geqslant k$. We may assume that $V_{1:k}$ is orthonormal with respect to $\tilde{S}$, i.e. $V_{1:k}^\dagger \tilde{S} V_{1:k} = I_k$, where I_k denotes the identity matrix of dimension k.
- Compute new approximations

$$Y_{\mathrm{new}} = [Y\ Z]V_{1:k} \tag{10}$$

  to the eigenvectors of (3).

Remarks 1.

- In the original Davidson method [5], the subspace is expanded according to

$$z_j = \mathrm{diag}^{-1}(H - \vartheta_j S)(H - \vartheta_j S)y_j, \quad j = 1, \ldots, k,$$

while in the full Jacobi–Davidson method [26], the subspace is expanded according to

$$\left(I_n - Sy_j y_j^\dagger\right)(H - \vartheta_j S)\left(I_n - y_j y_j^\dagger S\right)z_j = (H - \vartheta_j S)y_j. \tag{11}$$

The subspace expansion in (6a) corresponds to the first step of an iterative solution of the linear equations system (11), which would be far too expensive to solve. We replace ϑ_j by a scalar $\bar{\lambda} \in \mathbb{R}$ which should be close to the eigenvalues of interest (typically our eigenvalues are less than one). Numerical tests (see later) have shown that $\bar{\lambda}$ usually has little influence on the convergence and can also be set to zero, so that $(H_0 - \bar{\lambda}S_0)^{-1}$ can be replaced by H_0^{-1}.
- It is worth to mention that the preconditioner $(H_0 - \bar{\lambda}S_0)^{-1}$ (6b) is so efficient, that it needs to be calculated only once for a given basis set and can be kept constant throughout the scf-cycle, but even during optimization of internal atomic positions.

[1] This implies that there exists a basis of eigenvectors to the real eigenvalues $\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$ which are orthonormal w.r.t. the inner product defined by S.

- Note that in (6a) computing inverses of possibly singular matrices can easily be avoided by slightly perturbing $\bar{\lambda}$.
- If the scf-cycle is nearly converged, the initial guesses y_j may almost coincide with the corresponding exact eigenvectors of (3). In that case, the residual $(H-\vartheta_j S)y_j$ is zero, and the reduced system matrices $\tilde{H}$ and $\tilde{S}$ are singular, hence the reduced problem (7) admits no unique solution. This causes problems with the LAPACK routines used for the "small" (reduced) eigenvalue problem (7) in the parallel version. Our work-around to refine the remaining eigenpairs is to *project out* the converged eigenvalues and associated-vectors. Thus, if the index j corresponds to a converged state, then we set the associated entries $\tilde{S}_{j,i}=\tilde{H}_{j,i}:=0,\ \tilde{S}_{i,j}=\tilde{H}_{i,j}:=0,\ i=1,\ldots,m,\ \tilde{H}_{j,j}=10^8,\ \tilde{S}_{j,j}=1$. This moves the eigenvalue associated with the index j beyond the physically relevant range and thus this contribution no longer appears in the further computations.

3. Implementation details

The implementation of the iterative diagonalization scheme described above (Eqs. (3)–(5), (6a), (6b), (7)–(10)) is based on a highly optimized code using BLAS and LAPACK (or ScaLAPACK) routines. It can solve either a real symmetric or a complex hermitian eigenvalue problem (all in double precision) in sequential or (mpi-) parallel mode.

In order to set up the reduced eigenvalue problem (7) we first form the products of HY and SY (zhemm), then evaluate the upper part of the reduced Hamiltonian and overlap matrix $Y^\dagger HY, Y^\dagger SY$ (zher2k). The diagonals of these matrices are used to compute the Ritz values ϑ_j (5). This will be called $H1$ in Fig. 1.

The search space represented by the matrix Z defined in (6a) can be calculated in two different ways:

- as a solution of the linear system of equations:

$$(H-\bar{\lambda}S)z_j=(H-\vartheta_j S)y_j,\qquad j=1,\ldots,k \tag{12}$$

  which is solved (zhetrs) after factorization of $(H-\bar{\lambda}S)$ (zhetrf) in cases where the inverse cannot be stored on disk, or
- more efficiently, the inverse $(H_0-\bar{\lambda}S_0)^{-1}$ is calculated in the first scf-cycle (zhetri) and stored on disk (only one triangle of the matrix is actually stored in single precision to save disk space and speed up I/O). In all following scf-cycles it is read in and a simple matrix-matrix multiplication (zhemm) generates Z.

Formation of $\tilde{H}$ and $\tilde{S}$ according to (8) and (9) requires two more multiplications HZ and SZ (zhemm) and these three matrix-matrix multiplications are called $Z1$ in Fig. 1.

The products of HY and SY that enter the right-hand side of (8) and (9) have been obtained before and $\tilde{H}$ and $\tilde{S}$ can be evaluated using $Z^\dagger HY, Z^\dagger SY, Z^\dagger HZ$ and $Z^\dagger SZ$ (zgemm). The reduced eigenproblem (7), which is only of dimension $2\cdot k$, is usually solved quickly by full diagonalization as described in the introduction. The time spent on this part is call $H_{reduced}$ in Fig. 1.

4. Numerical results

We start this section by giving a decomposition of the total CPU time of the iterative scheme into the different tasks. At the first iterative step the inverse $(H_0-\bar{\lambda}S_0)^{-1}$ needs to be calculated and stored to disk. This is an αN^3operation like the full diagonalization, but the prefactor α is much smaller for inversion of a matrix [7] and thus this step is usually 2–4 times faster than full diagonalization. In all later scf-cycles (typically 20–200 cycles, in particular when also a structural optimization is performed) the inverse is read from disk, which even for the biggest test case takes only a few seconds.

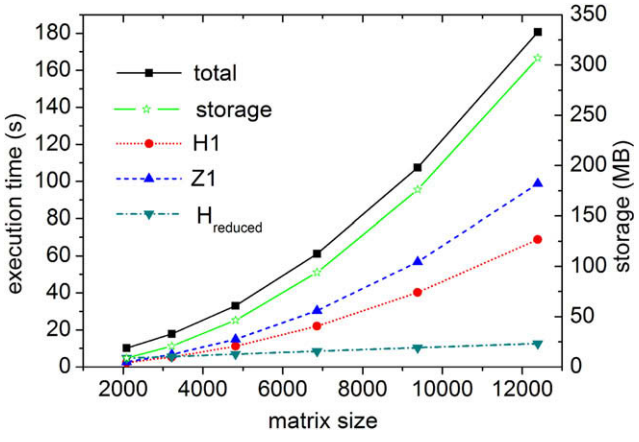

Fig. 1. CPU time analysis of our iterative diagonalization scheme of a real symmetric eigenvalue problem as function of matrix size and 600 eigenvalues (see text).

From Fig. 1 we can see that our iterative diagonalization scales nearly perfectly with N^2, similar to the amount of disk space (storage) of $(H_0 - \bar{\lambda}S_0)^{-1}$. Most of the time is spent on the multiplication of the H or S matrices with the Y or Z vectors, which scales as $N^2 \cdot k$ and our algorithm needs five such multiplications: HY, SY, HZ, SZ and a product forming Z. It is evident from Fig. 1 that the CPU time spent on two products HY, SY and two rank 2k updates ($H1$), as well as forming Z, HZ and SZ (three products, $Z1$) is usually the dominating part. The setup of $\tilde{H}$ and $\tilde{S}$ (8) and (9) and the solution of the reduced eigenvalue problem (7) is also indicated in Fig. 1. For small basis sets and a large number of eigenvalues this can be a significant fraction of the total time, but for realistic cases the effort proportional to $(2 \cdot k)^3$ is always small.

As discussed above, the CPU time of our iterative algorithm scales proportionally to $\alpha(N^2 \cdot k)$, where N is the size of H and k is the number of evaluated eigenvalues. Since the full diagonalization scales proportionally to $\alpha' N^3$ the gain in the CPU time depends on the ratio of α and α' and on how many eigenvalues k are needed. The actual speedup is indicated in Fig. 2, where the ratio of the CPU time of full and iterative diagonalization is shown for three different values of the number of eigenvalues k (and as a function of the matrix size N). For a "realistic" ratio of 5% between k and N a speedup of 8 can easily be achieved. Comparison with other algorithms in the literature is difficult, but e.g. in [19] only a factor of 3.3 (see their Table 2 for example using 32 processors) has been reached.

As mentioned in the introduction, our ultimate goal is to solve the KS equations for a particular solid or molecule self-consistently. Thus, to evaluate the efficiency and usefulness of this iterative diagonalization one should in fact consider the gain in time for one discrete scf-cycle, i.e. not only the time for diagonalization but also for setting up the eigenvalue problem as well as other parts like the calculation of the electron density ρ and the potential v_{eff}. Furthermore, one has to verify that the iterative diagonalization scheme does not increase the number of scf-cycles that are necessary to converge ρ. We will demonstrate this on two very different examples below:

Since WIEN2k is a code utilizing a finite unit cell with 3-dimensional periodic boundary conditions, a calculation of a single molecule must utilize large cells so that the molecules are well separated from each other. This leads to a rather large number of APW basis functions. We have done such calculations for the CO molecule and such an example can be considered as extreme case where, on the one hand, the time for setting up the eigenvalue problem (H, S) is rather negligible compared to the diagonalization, because they scale like $N_{bas}^2 \cdot N_{atoms}$, where the number of basis functions N_{bas} is 2800 while N_{atoms} is just 2 for this example. The eigenvalue problem is complex hermitian, because there is no inversion symmetry present. On the other hand only one k-point in the Brillouin zone (BZ) needs to be considered, i.e. only one eigenvalue problem needs to be solved in every scf-cycle and thus for more difficult to evaluate DFT functionals than the one used here, there is a relatively large fraction of time spent in the calculation of the potential v_{eff}. Furthermore, the number of required eigenvalues ϵ_i will be extremely small (below 1% of the matrix size). In fact as can be seen from Table 1 about 88% of the time goes into the diagonalization. Using our new iterative scheme, the diagonalization time can be reduced by a factor of 30, so that the eigenvalue problem becomes only a small fraction of the total time of one scf-cycle. As noted above, it is important that an iterative diagonalization scheme does not increase the number of scf-cycles too much, otherwise the gain in speed may be lost in the additional cycles required to reach self-consistency. In Fig. 3 we show the charge distance (the integral $|\rho_{in} - \rho_{out}|$) as function of the number of scf-iterations. The CO molecule has 5 occupied eigenstates and when we calculate just 8 states (i.e. only 3 unoccupied states), the number of cycles would increase drastically and there would be only a rather small gain in time. This case with such a limited search space is also one of the few examples we have found, where the value of $\bar{\lambda}$ has a large influence on the quality of the solution. Changing $\bar{\lambda}$ from 0 (this implies using just H^{-1} as preconditioner) to $\bar{\lambda} = -1$ (for a molecule in a big box the highest occupied eigenvalue is around -0.5 Ry, while for bulk calculations it is usually around +0.5 Ry) brings the number of scf-cycles back to a normal value. On the other hand, with 16 or 32 eigenvalues the required

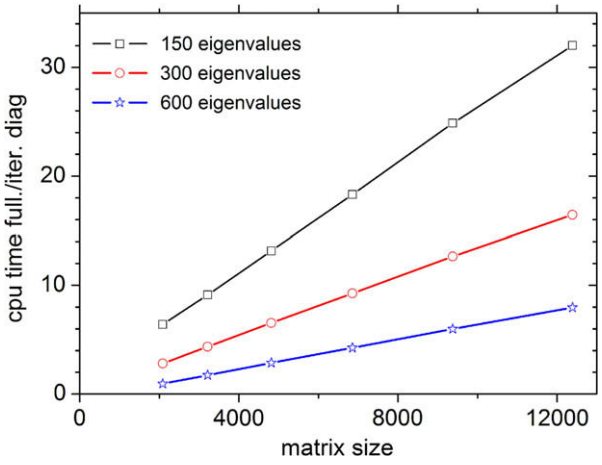

Fig. 2. Ratio of the CPU time of full and iterative diagonalization as function of matrix size for three different numbers of eigenvalues.

Table 1
Timings of different parts (see text) of the calculations for a CO molecule. The scf-timings take into account the required number of iterations to reach a convergence of 5×10^{-5} for the charge distance.

Task	Time (s)	iterations ($\bar{\lambda} = 0/-1$)	Time scf ($\bar{\lambda} = 0/-1$) (s)
ρ, v_{eff}	5.0	–	–
H, S	3.9	–	–
Full diag.	67.3	9	686
Iter. diag (8 ϵ_i)	1.6	31/13	426/237
Iter. diag (16 ϵ_i)	2.1	13/12	243/231
Iter. diag (32 ϵ_i)	2.9	13/12	253/241

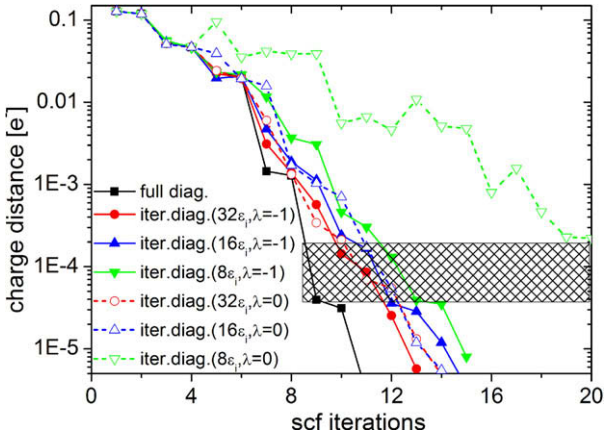

Fig. 3. Charge distance (integral $|\rho_{in} - \rho_{out}|$) as function of the number of scf-iterations for the CO molecule using full or iterative diagonalization (with 8, 16 or 32 eigenvalues and $\bar{\lambda} = 0$ and -1). The hatched region indicates the typically desired convergence criterion.

number of scf-cycles does not increase significantly and a large saving of cpu-time by about a factor of three can be obtained (see Table 1).

As a representative example of a typical medium-sized bulk calculation, we next present results for Stibnite, Sb_2S_3, a compound with 20 atoms/cell and inversion symmetry, thus the matrix elements are real numbers with $N_{bas} = 2400$. Typically one has to use about 10 k-points in the BZ, i.e. we have to solve 10 different eigenvalue problems in every scf-cycle, which has been taken into account for the corresponding scf-timings listed in Table 2. Setting up the eigenvalue problem is here a significant time (36% for full diagonalization) and it clearly dominates when the iterative diagonalization is used. The gain in speed is not so large, because of the relatively small matrix and the fair amount of eigenvalues (about 10%) required in this case, but still the total cpu time of the scf-procedure can be reduced by more than 40%. The convergence of the scf-cycles depends only mildly on the number of eigenvalues and a number typically twice the number of occupied bands (96 in this case) is more than sufficient (see Fig. 4).

For large scale calculations (more than 64 atoms/unit cell) the dimension of the eigenvalue problem becomes so large that it will either take very long time on a single processor or eventually the matrices cannot even be stored in memory on a single node. Thus, fine grained, mpi-based parallelization distributing the matrices over many processors and using ScaLAPACK routines [10] for linear algebra operations is necessary. In the following we present results for a 3×3 super-cell of a h-BN/Ni(1 1 1) surface model, which contains 99 atoms/cell. Since this is a magnetic and metallic system, 8 k-points and a spin-polarized calculation (i.e. 16 different eigenvalue problems per scf-cycle) had to be performed. The basis set size $N_{bas} = 16{,}900$ (real matrix elements) is so large, that we could not run this example on a single cpu, but the basic timings

Table 2
Timings of different parts (see text) of the calculations for Sb_2S_3. The scf-timings take into account 10 k-points and the required number of iterations to reach a convergence of 5×10^{-5} for the charge distance.

Task	Time (s)	Iterations	Time scf (s)
ρ, v_{eff}	35.0	–	–
H, S	8.8	–	–
Full diag.	15.8	13	3523
Iter. diag (150 ϵ_i)	2.7	16	2379
Iter. diag (216 ϵ_i)	3.7	15	2379
Iter. diag (361 ϵ_i)	6.4	13	2403

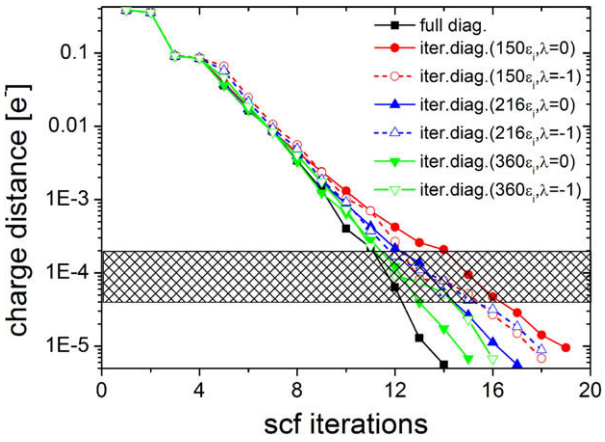

Fig. 4. Charge distance (integral vert$\rho_{in} - \rho_{out}$|) as function of the number of scf-iterations for Sb_2S_3 using full or iterative diagonalization (with 150, 216 or 361 eigenvalues). The hatched region indicates the typically desired convergence criterion.

Table 3
Timings on 4-processors of different parts (see text) of the calculations for a 3 × 3 super-cell of a h-BN/Ni(1 1 1) surface model with 99 atoms/cell (N_{bas} = 16900). The scf-timings assume 16 k-points.

Task	Time (s)
ρ, v_{eff}	125
H, S	1080
Full diag.	2850
Iter. diag (950 ϵ_i)	250
One scf-cycle (full diag.)	64,050
One scf-cycle (iter. diag. 950 ϵ_i)	22,450

will be given for 4 cpus (cluster of AMD dual-core opterons with infiniband network). For the iterative diagonalization we used 950 eigenvalues (about 600 are occupied) and virtually no scf-convergence degradation was found. As evident from Table 3, the time spent in calculating ρ and v_{eff} is almost negligible, but 70% of the time is spent during full diagonalization. The iterative scheme is more than ten times faster and thus only 25% of the time is spent on the diagonalization. Overall the time of an scf-cycle can be reduced by a factor of three. Still the cpu time in particular for the setup of H and S is so large that further parallelization is desirable and in Fig. 5 the scaling of setup and diagonalization of this eigenvalue problem up to 64 cpus is presented. It should be noted that these measurements were done under full-load conditions of the machine, which leads to some irregularities in the reported timings. The setup of the matrix elements is split into a spherical part (H_{sph},

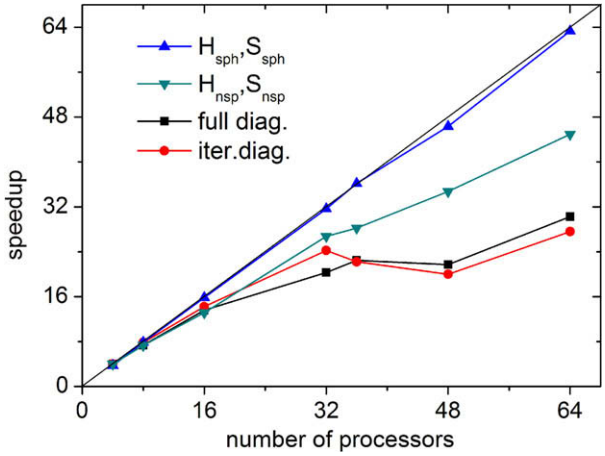

Fig. 5. Speedup of various parts of the eigenvalue problem with the number of processors for a 3 × 3 super-cell of a h-BN/Ni(1 1 1) surface model with 99 atoms/cell (N_{bas} = 16,900).

S_{sph}), which takes about 2/3 of the total setup time and a non-spherical part (H_{nsp} and S_{nsp}). H_{sph}, S_{sph} scales perfectly as it has basically no communication and also no sequential part. The calculation of the non-spherical matrix elements H_{nsp} and S_{nsp} still scales very well, but as one cannot avoid some sequential steps, decreasing efficiency is to be expected for more cpus. For up to 32 cpus the iterative diagonalization scales slightly better than full diagonalization (matrix–matrix multiplication vs. solution of eigenvalue problem), but then the overhead of the network based I/O and problems due to the stacked infiniband switch degrades the performance. Nevertheless, even for 64 cpus the iterative diagonalization is 10 times faster than full diagonalization and comparable to the setup of H_{sph}, S_{sph} and H_{nsp} and S_{nsp}, respectively. The efficiency (speedup/number of processors) of the full (iterative) diagonalization is 63 (75)% for 32 processors and 47 (43)% for 64 processors.

5. Conclusions

When one needs to solve the KS equations self-consistently, one has to solve a sequence of closely related generalized eigenvalue problems thus an iterative solution could lead to a significant speed-up. We propose a new preconditioner for the blocked Davidson method to solve efficiently generalized eigenvalue problems also for the case of non-diagonally dominant matrices. Commonly the inverse of the diagonal of $(H - \vartheta_j S)$ is used as preconditioner, but we use the inverse of $(H - \bar{\lambda} S)$ for a fixed value $\bar{\lambda}$ instead. The algorithm has been implemented in both, a sequential and a parallel version of our WIEN2k code and we demonstrate that it leads to a fast, stable and efficient scheme. In addition the new iterative scheme should show better scaling properties than the full diagonalization, at least as long as network I/O does not limit its performance. Most importantly, the approximate solution of the eigenvalue problem does not affect the solution of the self-consistent-field problem (i.e. the number of scf-iterations) provided a sufficient number of states is included.

This scheme has already been tested on an everyday basis by the large WIEN2k community [11] for several month and allowed us to increase the size of a materials science problem by a factor of up to two. Recently we could solve a big nano-science problem with more than 1100 atoms/unit cell [6,16], which is metallic and contains many transition metal atoms.

Acknowledgments

This work was supported by the Austrian Grid Project (WP 15).

References

[1] C. Bendtsen, O. Nielsen, L. Hansen, Solving large nonlinear generalized eigenvalue problems from density functional theory calculations in parallel, Appl. Numer. Math. 37 (2001) 189–199.
[2] P. Blaha, K. Schwarz, G. Madsen, Electronic structure calculations of solids using the WIEN2k package for material sciences, Comput. Phys. Commun. 147 (2002) 71–76.
[3] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka, J. Luitz, An augmented plane wave plus local orbital program for calculating crystal properties, 2001. ISBN 3-9501031-1-2.
[4] M. Crouzeix, B. Philippe, M. Sadkane, The Davidson method, SIAM J. Sci. Comput. 15 (1994) 62–76.
[5] E. Davidson, The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices, J. Comput. Phys. 17 (1975) 87–94.
[6] H. Dil, J. Lobo-Checa, R. Laskowski, P. Blaha, S. Berner, J. Osterwalder, T. Greber, Surface trapping of atoms and molecules with dipole rings, Science 319 (2008) 1824.
[7] G.H. Golub, C.F. Van Loan, Matrix Computations, second ed., The John Hopkins University Press, Baltimore, 1989.
[8] P. Hohenberg, W. Kohn, Inhomogeneous electron gas, Phys. Rev. 136 (1964) B864.
[9] <http://www.netlib.org/lapack/>.
[10] <http://www.netlib.org/scalapack/>.
[11] <http://www.wien2k.at>.
[12] W. Kohn, L.J. Sham, Self-consistent equations including exchange and correlation effects, Phys. Rev. 140 (1965) A1133.
[13] G. Kresse, J. Furthmüller, Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set, Comput. Mater. Sci. 6 (1996) 15–50.
[14] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, Phys. Rev. B 54 (1996) 11169–11186.
[15] G. Kresse, J. Furthmüller, VASP the GUIDE, 2007, <http://cms.mpi.univie.ac.at/VASP>.
[16] R. Laskowski, P. Blaha, Unraveling the structure of the h-bn/rh(1 1 1) nanomesh with ab initio calculations, J. Phys.: Condens. Matter 20 (2008) 064207.
[17] G.K.H. Madsen, P. Blaha, K. Schwarz, E. Sjöstedt, L. Nordström, Efficient linearization of the augmented plane-wave method, Phys. Rev. B 64 (2001) 195134.
[18] P. Pulay, Convergence acceleration of iterative sequences. The case of SCF iteration, Chem. Phys. Lett. 73 (1980) 393.
[19] M.J. Rayson, P.R. Briddon, Rapid iterative method for electronic-structure eigenproblems using localized basis functions, Comput. Phys. Commun. 178 (2008) 128–134.
[20] K. Schwarz, DFT calculations of solids with LAPW and WIEN2k, Solid State Commun. 176 (2003) 319–328.
[21] K. Schwarz, P. Blaha, Solid state calculations using WIEN2k, Comput. Mater. Sci. 28 (2003) 259–273.
[22] D. Singh, Simultaneous solution of diagonalization and self-consistency problems for transition-metal systems, Phys. Rev. B 40 (1989) 5428–5431.
[23] G.L.G. Sleijpen, A.G.L. Booten, D.R. Fokkema, H.A. Van der Vorst, Jacobi–Davidson type methods for generalized eigenproblems and polynomial eigenproblems, BIT 36 (3) (1996) 595–633.
[24] G.L.G. Sleijpen, H.A. Van der Vorst, A Jacobi–Davidson iteration method for linear eigenvalue problems, SIAM J. Matrix Anal. Appl. 17 (2) (1996) 401–425.
[25] M.P. Teter, M. Payne, D.C. Allan, Solution of Schrödinger's equation for large systems, Phys. Rev. B 40 (18) (1989) 12255.
[26] H.J.J. van Dam, J.H. van Lenthe, G.L.G. Sleijpen, H.A. van der Vorst, An improvement of Davidson's iteration method, J. Comput. Chem. 17 (3) (1996) 267–272.
[27] J. VandeVondele, J. Hutter, An efficient orbital transformation method for electronic structure calculations, J. Chem. Phys. 118 (10) (2003) 4365–4369.
[28] D.M. Wood, A. Zunger, A new method for diagonalising large matrices, J. Phys. A: Math. Gen. 18 (1985) 1343–1359.