Computer Physics Communications 126 (2000) 294–309

Computer Physics Communications

www.elsevier.nl/locate/cpc

Improving the efficiency of FP-LAPW calculations

Max Petersen [a,*], Frank Wagner [a], Lars Hufnagel [a], Matthias Scheffler [a], Peter Blaha [b], Karlheinz Schwarz [b]

[a] *Fritz-Haber-Institut der Max-Planck-Gesellschaft, Faradayweg 4-6, D-14 195 Berlin (Dahlem), Germany*
[b] *Institut f. Physikalische und Theoretische Chemie, Technische Universität Wien, Getreidemarkt 9/156, A-1060 Vienna, Austria*

Received 12 January 1999; received in revised form 25 June 1999

Abstract

The *full-potential linearized augmented-plane wave* (FP-LAPW) method is well known to enable most accurate calculations of the electronic structure and magnetic properties of crystals and surfaces. The implementation of atomic forces has greatly increased its applicability, but it is still generally believed that FP-LAPW calculations require substantial higher computational effort compared to the pseudopotential plane wave (PPW) based methods.

In the present paper we analyze the FP-LAPW method from a computational point of view. Starting from an existing implementation (WIEN95 code), we identified the time consuming parts and show how some of them can be formulated more efficiently. In this context also the hardware architecture plays a crucial role. The remaining computational effort is mainly determined by the setup and diagonalization of the Hamiltonian matrix. For the latter, two different iterative schemes are compared. The speed-up gained by these optimizations is compared to the runtime of the "original" version of the code, and the PPW approach. We expect that the strategies described here, can also be used to speed up other computer codes, where similar tasks must be performed.

PACS: 02.60.Pn; 71.15.Mb; 71.15.Ap

PROGRAM SUMMARY

Title of program extension: `wien-speedup`

Catalogue identifier: ADLP

Program Summary URL: http://cpc.cs.qub.ac.uk/summaries/ADLP

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Other version of the program: cat. no.: ABRE; *title:* `WIEN`; *ref. in CPC:* 59 (1990) 399

Licensing provisions: none

Computer, operating system, and installation:
IBM RS/6000; AIX; Fritz-Haber-Institut der Max-Planck-Gesellschaft; Berlin

Operating system: UNIX

Programming language: FORTRAN77

Unusual features of the program:
On IBM RS/6000 nodes part of the speedup was obtained by using an IBM specific mathematical library [1] (see Long Write-up, Section 4.4.1)

* Corresponding author. E-mail: petersen@FHI-Berlin.MPG.DE.

0010-4655/00/$ – see front matter
PII: S0010-4655(99)00495-6

Floating point arithmetic: 64 bits

Memory required to execute with typical data: 64 Mbyte (depends on case)

No. of bytes in distributed program, including test data, etc.: 3 355 093 bytes

Distribution format: uuencoded compressed tar

No. of bits in a word: 64

No. of processors used: one

Has the code been vectorized? no

Memory required for test run: 64 MByte

Keywords: Density-functional theory, linearized augmented plane wave method, LAPW, supercell, total energy, crystals, surfaces, molecules

Nature of the physical problem
For *ab-initio* studies of the electronic and magnetic properties of poly-atomic systems, such as molecules, crystals, and surfaces.

Method of solution
The full-potential linearized augmented plane wave (FP-LAPW) method is well known to enable accurate calculations of the electronic structure and magnetic properties of crystals [2–12]. Within the supercell approach it has also been used for studies of defects in the bulk and for crystal surfaces.

References
[1] http://www.rs6000.ibm.com/resource/technology/MASS/index.html.
[2] J.C. Slater, Phys. Rev. 51 (1937) 846; Adv. Quantum Chemistry 1 (1964) 35.
[3] H. Bross, Phys. Kondens. Mater. 3 (1964) 119; Z. Phys. B 81 (1990) 233.
[4] T.L. Loucks, Augmented Plane Wave Method (Benjamin, New York, 1967).
[5] D.D. Koelling, J. Phys. Chem. Solids 33 (1972) 1335; D.D. Koelling, G.O. Arbman, J. Phys. F 5 (1975) 2041.
[6] O.K. Andersen, Solid State Commun. 13 (1973) 133; Phys. Rev. B 12 (1975) 3060.
[7] E. Wimmer, H. Krakauer, M. Weinert, A.J. Freeman, Phys. Rev. B 24 (1981) 864.
[8] H.J.F. Jansen, A.J. Freeman, Phys. Rev. B 30 (1984) 561.
[9] L.F. Mattheiss, D.R. Hamann, Phys. Rev. B 33 (1986) 823.
[10] P. Blaha, K. Schwarz, P. Sorantin, S.B. Trickey, Comput. Phys. Commun. 59 (1990) 399.
[11] P. Blaha, K. Schwarz, R. Augustyn, WIEN93 (Technical University, Vienna, 1993); improved and updated UNIX version of the original copyrighted `WIEN`-code [10].
[12] D.J. Singh, Planewaves, Pseudopotentials and the LAPW Method (Kluwer Academic, Boston, 1994).

LONG WRITE-UP

1. Introduction

The augmented plane wave (APW) method [1,2,4–6] and in particular its linearized form, the LAPW method [7–15], enables accurate calculations of electronic and magnetic properties of poly-atomic systems using density-functional theory (DFT) [16,17]. One successful implementation of the full-potential LAPW (FP-LAPW) method is the program package `WIEN`, a code developed by Blaha, Schwarz and coworkers [14]. It has been successfully applied to a wide range of problems such as electric field gradients [18,19] and systems such as high-temperature superconductors [20], minerals [21], surfaces of transition metals [22], or anti-ferromagnetic oxides [23] and even molecules [24]. Minimizing the total energy of a system by relaxing the atomic coordinates for complex systems became possible by the implementation of atomic forces [24], and even molecular dynamics became feasible. Up to now the main drawback of the FP-LAPW-method compared to the pseudopotential plane-wave (PPW) (e.g., Ref. [25] and references therein) approach has been its higher computational expense. This may be mainly due to a discrepancy in optimization efforts spent on both methods, and therefore we have analyzed the FP-LAPW method from a computational/numerical point of view. Starting from the `WIEN95` implementation [26], we identified the time consuming parts and will show how some of them can be formulated more efficiently. In this context also the influence of the underlying hardware architecture will be discussed.

The remainder of the paper is organized as follows. After introducing the principles of DFT and summarizing the concepts of the FP-LAPW-method (Sections 2 and 3), we will report on our improvements made on the `WIEN95` implementation of the FP-LAPW-method (Section 4). In Section 5 we will show, how these improvements make the FP-LAPW-method a strong competitor to the popular PPW approach by comparing the run-times necessary to converge a nine layer slab of (4×2)-Cu(110) (i.e. 72 atoms and 792 valence electrons) using both methods.

2. Density-functional theory

The central statement of DFT is, that the problem of finding the ground-state energy of a many-particle system, characterized by a many-particle wavefunction Ψ_0, can be mapped on a physically equivalent problem of finding the ground-state electron density n_0, i.e.

$$E[\Psi_0] = E[n_0] \tag{1}$$

with

$$n_0(\mathbf{r}) = \left\langle \Psi_0 \middle| \sum_{\alpha}^{N} \delta(\mathbf{r} - \mathbf{r}_\alpha) \middle| \Psi_0 \right\rangle, \tag{2}$$

where $\mathbf{r}_\alpha$ is the coordinate of the αth electron. The central statement of the Hohenberg–Kohn theorem [16] is, that for an N electron system the functional $E[n]$ is minimized by the ground-state electron density, n_0.

$$E[n_0] = \operatorname{Min} E[n] \tag{3}$$

with the constraint,

$$\int n \, \mathrm{d}^3 r = N. \tag{4}$$

In the Kohn–Sham formulation the functional $E[n]$ is split into the following terms:

$$E[n] = T_\mathrm{s}[n] + U[n] + E_\mathrm{xc}[n], \tag{5}$$

the kinetic energy functional of non-interacting particles, $T_\mathrm{s}[n]$, the functional of the electrostatic energy, $U[n]$ and the rest, called exchange-correlation energy, $E_\mathrm{xc}[n]$. With Eq. (5), i.e. with the introduction of the functional $T_\mathrm{s}[n]$, the variational problem of Eqs. (3), (4) becomes equivalent to the problem of solving a system of single-particle equations, called the Kohn–Sham equations [17],

$$H\varphi_i = \left[-\frac{\hbar^2}{2m_\mathrm{e}} \nabla^2 + V_\mathrm{eff} \right] \varphi_i = \varepsilon_i \varphi_i, \tag{6}$$

$$n = \sum_i f_i \varphi_i^* \varphi_i. \tag{7}$$

Here, $-\frac{\hbar^2}{2m_\mathrm{e}} \nabla^2$ is the single-particle kinetic energy operator and V_eff is the potential defined by the functional derivative of $U[n] + E_\mathrm{xc}[n]$,

$$V_\mathrm{eff} = \frac{\delta(U + E_\mathrm{xc})}{\delta n}. \tag{8}$$

The electron density is obtained from Eq. (7), where f_i are the occupation numbers given by the Fermi distribution. In practice Eqs. (6)–(8) are solved in a selfconsistent field (SCF)-cycle: i.e. starting with density n_1 one calculates the potential V_eff, solves Eq. (6) and by evaluating Eq. (7) one obtains the new density n_2, which leads to the next iteration cycle.

3. The FP-LAPW-method

In the augmented plane-wave (APW) method space is divided into an interstitial region (IR) and non-overlapping muffin-tin (MT) spheres centered at the atomic sites [1]. This allows an accurate description of both, the rapidly changing (oscillating) wavefunctions, potential and electron density close to the nuclei as well as the smoother part of these quantities in between the atoms. In the IR the basis set consists of plane waves $\exp(i\mathbf{K}\cdot\mathbf{r})$. The choice of a computationally efficient and accurate representation of the wavefunctions within the MT spheres has been discussed by several authors, e.g., [4,7,8,10]. In the original APW formulation introduced by Slater [1,2], the plane-waves are augmented to the exact solutions of the Schrödinger equation within the MT at the calculated eigenvalues. This approach is computationally expensive because it leads to an explicit energy dependence of the basis functions (and consequently of the Hamilton- and overlap-matrices) and thus to a non-linear eigenvalue problem. Instead of performing a single diagonalization to solve the KS equation one repeatedly needs to evaluate (for many trial energies) the determinant of the secular equation in order to find its zeros and thus the single particle eigenvalues ε_i. Going into the complex energy plane would have been one option but was not explored so far, except in an other context (see, e.g., [3] and references therein).

In the linearized APW method (LAPW) the problem of the energy dependence of the basis set is removed by using a fixed set of suitable MT radial functions [7,8,10]. Within Andersen's approach, used also in the `WIEN` code, inside each atomic sphere I and for azimuthal quantum number l the radial solutions $u_l^I(\varepsilon_l^I, r_I)$ of the KS equation at fixed energies ε_l^I and their energy derivatives $\dot{u}_l^I(\varepsilon_l^I, r_I)$ are used as basis functions. Basically, this choice corresponds to a linearization of the energy dependence of $u_l^I(\varepsilon, \mathbf{r})$ around ε_l^I [10]. The concept implies that the radial functions $u_l^I(\varepsilon_l)$ and $\dot{u}_l^I(\varepsilon_l)$ and the respective overlap and Hamilton matrix elements need to be calculated only for a few energies ε_l^I. Moreover, all KS energies ε_i are found, for each $\mathbf{k}$-point, by only one diagonalization (for a detailed discussion see [15]).

The LAPW basis functions $\phi_{\mathbf{G}}(\mathbf{r},\mathbf{k})$ which are used for the expansion of the KS wavefunctions

$$\psi_i(\mathbf{r},\mathbf{k}) = \sum_{|\mathbf{k}+\mathbf{G}|\leqslant G^{\mathrm{wf}}} c_i(\mathbf{k}+\mathbf{G})\phi_{\mathbf{G}}(\mathbf{r},\mathbf{k}) \tag{9}$$

are defined as

$$\phi_{\mathbf{G}}(\mathbf{r},\mathbf{k}) = \begin{cases} \Omega^{-1/2}\exp\bigl(i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}\bigr), & \mathbf{r}\in \mathrm{IR}, \\ \sum_I \sum_{lm}\bigl[a_{lm}^I(\mathbf{k}+\mathbf{G})\,u_l^I(\varepsilon_l^I, r_I) + b_{lm}^I(\mathbf{k}+\mathbf{G})\,\dot{u}_l^I(\varepsilon_l^I, r_I)\bigr]Y_{lm}(\hat{\mathbf{r}}_I), & r_I \leqslant s_I. \end{cases} \tag{10}$$

Here, $\mathbf{G}$ denote the reciprocal lattice vectors and $\mathbf{k}$ a vector within the first Brillouin zone. The wave function cutoff G^{wf} limits the number of the $\mathbf{G}$ vectors and thus the size of the basis set. The symbols in Eq. (10) have the following meaning: Ω is the unit cell volume, s_I is the MT radius, and $\mathbf{r}_I = \mathbf{r} - \mathbf{R}_I$ is a vector within the MT sphere of the Ith atom. Note that $Y_{lm}(\hat{\mathbf{r}})$ represents a complex spherical harmonic with $Y_{l-m}(\hat{\mathbf{r}}) = (-1)^m Y_{lm}^*(\hat{\mathbf{r}})$. The radial functions $u_l(\varepsilon_l, r)$ and $\dot{u}_l(\varepsilon_l, r)$ are solutions of the equations

$$H^{\mathrm{sph}}\,u_l(\varepsilon_l, r) = \varepsilon_l\,u_l(\varepsilon_l, r), \tag{11}$$

$$H^{\mathrm{sph}}\,\dot{u}_l(\varepsilon_l, r) = \bigl[\varepsilon_l \dot{u}_l(\varepsilon_l, r) + u_l(\varepsilon_l, r)\bigr], \tag{12}$$

which are regular at the origin. The operator H^{sph} contains only the spherical average, i.e. the $l = 0$ component, of the effective potential within the MT. The ε_l should be chosen near the center of the energy band with the corresponding l-character. The coefficients $a_{lm}(\mathbf{k}+\mathbf{G})$ and $b_{lm}(\mathbf{k}+\mathbf{G})$ are determined by requiring that value and slope of the basis functions are continuous at the surface of the MT sphere.

The representation of the potential and electron density resembles the one employed for the wave functions, i.e.

$$n^{\text{eff}}(\mathbf{r}) = \begin{cases} \sum_I \sum_{lm} n^{\text{eff}}_{lm,I}(r_I) Y_{lm}(\hat{\mathbf{r}}_I), & r_I \leqslant s_I, \\ \sum_{|\mathbf{G}| \leqslant G^{\text{pot}}} n^{\text{eff}}_{\mathbf{G}} \exp(i\mathbf{G}\cdot\mathbf{r}), & \mathbf{r} \in \text{IR}. \end{cases} \tag{13}$$

Thus, no shape approximation is introduced and therefore such an approach is called a full-potential treatment. The quality of this description is controlled by the cutoff parameter G^{pot} for the lattice vectors $\mathbf{G}$ and the number of the (l, m)-terms included inside the MTs.

4. Improving the WIEN code

4.1. Optimization strategies

To achieve an optimal performance of a computer code on modern computers, it is essential that the used algorithms match the underlying hardware architecture. On todays computers, often the memory bandwidth is the limiting factor, i.e. the floating-point operation units are stalled, waiting for data. Then the performance is not determined by the number of floating point operations per second, but by the necessary number of load/store operations. Therefore a significant objective of optimizing a code is to reduce the communication between the processor and the relatively slow memory, but to make optimum use of the fast cache. Thus the well known fact for parallel computers, that an efficient use of communication is crucial for complex and time consuming calculations, holds also on stand-alone workstations. The best way to improve the performance of a program on a wide range of architectures without loosing portability, is to write the code in such a way that the bulk of the calculations is performed by calls to the well known *basic linear algebra subprograms* (*BLAS*) [27,28]; efficiency can then be obtained by using optimized implementations of these routines, specifically tailored to the hardware used. While on vector machines, the so-called Level 2 *BLAS* routines (matrix–vector-operations) lead to very satisfactory results, this approach is often not well suited for architectures of modern high-performance workstations or shared memory systems with a hierarchy of memory (registers, cache, local memory, swap space). For those architectures it is preferable to partition the matrices into blocks and to perform the computation by matrix–matrix-operations on these blocks. This leads to a full reuse of data already held in cache (or local memory) and reduces data movement. While for Level 1 (vector–vector-operations) and Level 2 (vector–matrix-operations) *BLAS* routines the number of load/store operations is proportional to the number of floating-point operations, the Level 3 (matrix–matrix-operations) approach [29] gives a surface-to-volume effect, i.e. if the matrices are of order n, the number of floating-point operations is of order n^3, while the number of load/store operations is of order n^2. This minimizes the influence of a limited memory bandwidth on the performance of the program. Therefore the goal in optimizing the code must be to use Level 3 *BLAS* routines as much as possible.

4.2. The structure of the WIEN-code

The SCF cycle of the WIEN code consist of five independent programs:

(1) LAPW0: generates the potential from a given charge density
(2) LAPW1: computes the eigenvalues and eigenvectors
(3) LAPW2: computes the valence charge density from the eigenvectors
(4) CORE: computes the core states and densities
(5) MIXER: mixes the densities generated by LAPW2 and CORE with the density of the previous iteration to generate a new charge density

From these programs LAPW1 and LAPW2 are the most time consuming, while the time needed to run CORE and MIXER are basically negligible. Further inspection showed, that for example on IBM RS/6000 nodes the performance of LAPW2 was far below the theoretical peak performance, which indicates a poor adaptation of the code to this hardware architecture. The optimizations done on LAPW2 are described in Section 4.3. The situation was different in the case of LAPW1. Due to the use of standard library routines the diagonalization of the matrix, which is the most time consuming part, performs quite well on IBM RS/6000 nodes. However, on several other hardware platforms with substantially slower memory bandwidth, the performance was not so good and those routines were also modified to increase performance. Thus further improvement on IBM RS/6000 could only be reached by implementing a new algorithm. Based on the fact that the matrix to be diagonalized changes only little from iteration to iteration during the selfconsistency cycle, an iterative diagonalization scheme could be an attractive alternative. We implemented two such schemes, which use the information from the previous step to speed up the diagonalization. The details will be described in Section 4.4.

4.3. LAPW2: *Generating the electron density*

In LAPW2 the eigenvalues and eigenvectors found by LAPW1 are read in. The **k**-space integration over the Brillouin zone (BZ) is replaced by a finite **k**-summation, in which each **k**-point contributes with a weight, $W_j(\mathbf{k})$, in which for convenience also the occupation factor of state ε_j (i.e. the Fermi factor) is stored. First the Fermi energy and then the expansion of the valence electron density is calculated for each of the occupied states at all **k**-points in the irreducible part of the BZ. The valence electron density consists of two types of components: the electron density inside each sphere I, $n^I(\mathbf{r})$, represented in spherical harmonics on a radial grid and the interstitial electron density, $n^{IR}(\mathbf{r})$ expressed as Fourier series.

4.3.1. The electron density inside the MT-spheres

The valence electron density inside a sphere is given by the expression:

$$n^I(\mathbf{r}) = \sum_{l''m''} n^{\text{eff}}_{l''m'',I}(r_I)\, Y_{LM}(\hat{\mathbf{r}}_I) \quad r_I \leqslant s_I \tag{14}$$

$$\begin{aligned} &= \sum_{\mathbf{k},j} W_j(\mathbf{k}) \sum_{lm} \sum_{l'm'} \sum_{\mathbf{G},\mathbf{G}'} \big\{ c^*(j,\mathbf{k}+\mathbf{G})\, a^{I*}_{lm}(\mathbf{k}+\mathbf{G})\, u_l(r)\, c(j,\mathbf{k}+\mathbf{G}')\, a^I_{l'm'}(\mathbf{k}+\mathbf{G}')\, u'_l(r) \\ &\quad + c^*(j,\mathbf{k}+\mathbf{G})\, b^{I*}_{lm}(\mathbf{k}+\mathbf{G}) \dot{u}_l(r)\, c(j,\mathbf{k}+\mathbf{G}')\, a^I_{l'm'}(\mathbf{k}+\mathbf{G}')\, u'_l(r) \\ &\quad + c^*(j,\mathbf{k}+\mathbf{G})\, a^{I*}_{lm}(\mathbf{k}+\mathbf{G})\, u_l(r)\, c(j,\mathbf{k}+\mathbf{G}')\, b^I_{l'm'}(\mathbf{k}+\mathbf{G}') \dot{u}'_l(r) \\ &\quad + c^*(j,\mathbf{k}+\mathbf{G})\, b^{I*}_{lm}(\mathbf{k}+\mathbf{G}) \dot{u}_l(r)\, c(j,\mathbf{k}+\mathbf{G}')\, b^I_{l'm'}(\mathbf{k}+\mathbf{G}') \dot{u}'_l(r) \big\} Y^*_{lm}(\hat{\mathbf{r}}) Y_{l'm'}(\hat{\mathbf{r}}). \end{aligned} \tag{15}$$

With the definition

$$A^I_{lmj}(\mathbf{k}) := \sum_{\mathbf{G}} c(j,\mathbf{k}+\mathbf{G})\, a^I_{lm}(\mathbf{k}+\mathbf{G}), \tag{16}$$

$$B^I_{lmj}(\mathbf{k}) := \sum_{\mathbf{G}} c(j,\mathbf{k}+\mathbf{G})\, b^I_{lm}(\mathbf{k}+\mathbf{G}), \tag{17}$$

the electron density reads:

$$\begin{aligned} n^I(\mathbf{r}) = \sum_{\mathbf{k},j} W_j(\mathbf{k}) \sum_{lm} \sum_{l'm'} \big\{ & A^{I*}_{lmj}(\mathbf{k}) A^I_{l'm'j}(\mathbf{k}) u_l(r) u_{l'}(r) \\ & + B^{I*}_{lmj}(\mathbf{k}) A^I_{l'm'j}(\mathbf{k}) \dot{u}_l(r) u_{l'}(r) + A^{I*}_{lmj}(\mathbf{k}) B^I_{l'm'j}(\mathbf{k}) u_l(r) \dot{u}_{l'}(r) \\ & + B^{I*}_{lmj}(\mathbf{k}) B^I_{l'm'j}(\mathbf{k}) \dot{u}_l(r) \dot{u}_{l'}(r) \big\} Y^*_{lm}(\hat{r}) Y_{l'm'}(\hat{\mathbf{r}}). \end{aligned} \tag{18}$$

It is obvious that the calculation of the sums (16), (17) which run over all **G**-vectors for every combination of (I, j, lm), will be the most time consuming part, and thus needs special care to implement it efficiently. The straight forward implementation of the summation, as done in the original `WIEN` code, results in a high ratio of load/store operations per floating-point operation and a very poor performance. A closer look shows that these formulas can be rewritten in the form of a matrix–matrix-multiplication:

$$A^{I,\mathbf{k}}(j,lm) = \sum_{\mathbf{G}} c(j,\mathbf{k}+\mathbf{G})\, a^I(\mathbf{k}+\mathbf{G},lm), \tag{19}$$

$$B^{I,\mathbf{k}}(j,lm) = \sum_{\mathbf{G}} c(j,\mathbf{k}+\mathbf{G})\, b^I(\mathbf{k}+\mathbf{G},lm). \tag{20}$$

In this way the matrices $A^{I,\mathbf{k}}(j,lm)$ and $B^{I,\mathbf{k}}(j,lm)$ can be calculated using optimized (*BLAS-3*) library-routines, hereby reducing the number of load/store operations as well as minimizing the number of cache misses.

4.3.2. The interstitial electron density

The valence electron density in the interstitial region is given by:

$$n^{IR}(\mathbf{r}) = \sum_{|\mathbf{K}|\leqslant K^{\mathrm{pot}}} n^{\mathrm{eff}}_{\mathbf{K}}(\mathbf{r}) \exp(i\mathbf{K}\cdot\mathbf{r}), \quad \mathbf{r}\in \mathrm{IR} \tag{21}$$

$$= \sum_{\mathbf{k},j} \sum_{\mathbf{G}\mathbf{G}'} W_{\mathbf{k}}(j) c_{\mathbf{k}}(j,\mathbf{G}) c^*_{\mathbf{k}}(j,\mathbf{G}') \exp\bigl(i(\mathbf{G}-\mathbf{G}')\cdot\mathbf{r}\bigr), \tag{22}$$

where the sum over the occupied states j can again be regarded as matrix–matrix-multiplication (see Fig. 1), in which the matrix W consists of j identical columns $W_{\mathbf{k}}(j)$:

$$\tilde{n}_{\mathbf{k}}(\mathbf{G},\mathbf{G}') := \sum_j \underbrace{W_{\mathbf{k}}(j) c^T_{\mathbf{k}}(\mathbf{G},j)}_{\tilde{c}_{\mathbf{k}}(j,\mathbf{G})} c^*_{\mathbf{k}}(j,\mathbf{G}') \tag{23}$$

$$= \sum_j \tilde{c}^T_{\mathbf{k}}(\mathbf{G},j) c^*_{\mathbf{k}}(j,\mathbf{G}') \tag{24}$$

$$n(\mathbf{r}) = \sum_{\mathbf{k}} \sum_{\mathbf{G}\mathbf{G}'} \tilde{n}_{\mathbf{k}}(\mathbf{G},\mathbf{G}') \exp\bigl(i(\mathbf{G}-\mathbf{G}')\cdot\mathbf{r}\bigr). \tag{25}$$

Since $W_j(\mathbf{k})$ is real, the matrix $\tilde{n}(\mathbf{G},\mathbf{G}')$ is Hermitian, i.e. $\tilde{n}(\mathbf{G},\mathbf{G}') = \tilde{n}^*(\mathbf{G}',\mathbf{G})$. Therefore the calculation of the matrix

$$\tilde{n}_{\mathbf{k}}(\mathbf{G},\mathbf{G}') = \sum_j c^T_{\mathbf{k}}(\mathbf{G},j) c^*_{\mathbf{k}}(j,\mathbf{G}') \tag{26}$$

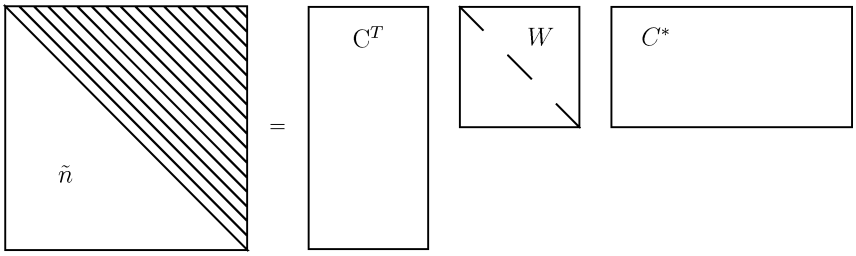

Fig. 1. The calculation of the interstitial electron-density in **k**-space can be regarded as matrix–matrix-multiplication $\tilde{n} = C^T W C^*$, where W consists of j identical vectors $W_{\mathbf{k}}(j)$.

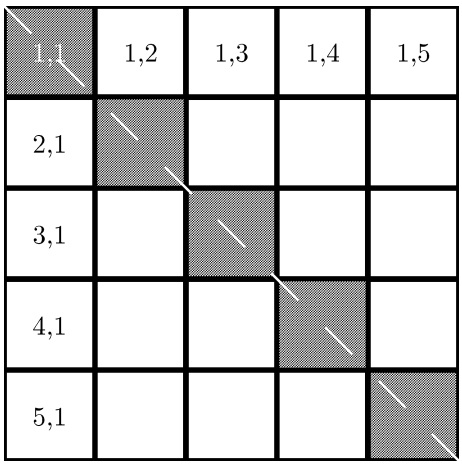

Fig. 2. The Hermitian matrix $\tilde{n}_{\mathbf{k}}(\mathbf{G}, \mathbf{G}')$ is divided into small blocks. Each of the blocks is calculated by a matrix–matrix-multiplication.

by a single matrix–matrix-multiplication would result in twice as much floating-point operations as necessary, which would destroy the advantage of using optimized library routines.

To profit from both, the hermiticity of the matrix and the use of optimized *BLAS-3* library routines, the matrix is divided into small blocks (Fig. 2). Each block above the diagonal is evaluated by a single *(BLAS-3)* matrix–matrix-multiplication according to Eq. (24) and the result is also used for the corresponding block below the diagonal. The elements of the blocks along the diagonal are evaluated by a direct implementation of the summation:

$$\tilde{n}_{\mathbf{k}}(\mathbf{G}, \mathbf{G}') = \tilde{n}^*_{\mathbf{k}}(\mathbf{G}', \mathbf{G}) = \sum_j c^T_{\mathbf{k}}(\mathbf{G}, j) c^*_{\mathbf{k}}(j, \mathbf{G}'), \quad \mathbf{G} \leqslant \mathbf{G}'. \tag{27}$$

The blocksize is a free parameter which has to be optimized according to the cache size of the specific platform.

4.4. LAPW1: *Setup and diagonalization of the eigenvalue problem*

4.4.1. Setup of H and S

According to Eq. (9) the KS eigenstates are characterized by a set of expansion coefficients $c_i(\mathbf{k} + \mathbf{G})$ $\{i = 1, \ldots, N_\mathrm{s}\}$, where N_s are the number of eigenstates to be calculated. In the following, these expansion coefficients are viewed as eigenvectors (of length N_pw) of the generalized eigenvalue problem

$$(H - \varepsilon_i S) c_i = 0, \tag{28}$$

where H is the Hamiltonian and S the overlap matrix. The elements of H and S are given by

$$H_{ij} = \langle \phi_i | H | \phi_j \rangle, \tag{29}$$

$$S_{ij} = \langle \phi_i | \phi_j \rangle, \tag{30}$$

where ϕ_j are the LAPW basis functions. As discussed earlier, one of the main ideas of the FP-LAPW method is to construct sophisticated basis functions φ which provide a good approximation to the true wave function ψ, so that the number of basis functions N_pw required to expand ψ with reasonable accuracy, is kept small. The main drawback of this approach is that the evaluation of Eqs. (29)–(30) is quite demanding. A simple way to reduce the computational effort in setting-up H is to consider in the first half of the self-consistency cycle only the spherical average of the potential (i.e. the $LM = (0, 0)$ component). Furthermore a considerable speedup on IBM RS/6000 nodes was obtained by using an IBM specific mathematical library [30] which allows a much faster evaluation of

trigonometric functions that are required in Eqs. (29)–(30). These subroutines compute the trigonometric functions for a vector of arguments, hereby minimizing the computational costs compared to the serial evaluation of all vector elements.

A combination of these procedures can significantly speed up the generation of H and S matrices.

4.4.2. Solving the eigenvalue problem

As noted before, the standard diagonalization routines could not be improved significantly on IBM RS/6000 nodes, since the modified LAPACK routines together with IBMt's highly optimized scientific ESSL library yields already almost optimal performance. On other hardware platforms (e.g., SGI Power Challenge, DEC-Alpha, Intel PII) with slower memory bandwidth we could achieve a speedup of the diagonalization by more than a factor of two by modifying the standard LAPACK routines using a hierarchical blocking scheme as described in [31].

4.4.3. Iterative diagonalization

In contrast to the LAPW method, the plane wave basis set used in the PPW method allows an easy evaluation of Eqs. (29)–(30), but the number of expansion functions is much larger. For this reason the approach to an iterative matrix diagonalization described below is somewhat different from the one usually adopted in the PPW method.

We implemented two schemes of iterative matrix diagonalization, namely the Block–Davidson and the Lanczos algorithm. As both methods are fairly well known, here only general aspects will be discussed, as far as they concern the FP-LAPW method. For a detailed discussion see, e.g., [32].

Since the KS equations must be solved self-consistently, the matrix C of the eigenvectors c_j in Eq. (28) are always available (with the exception of the first cycle) from a previous cycle, C^{old}. Therefore C^{old} can be used to obtain an approximate solution to Eq. (28). If H^{new} (S^{new}) is the Hamiltonian (overlap) matrix of the present iteration, then Eq. (28) can be transformed into the space spanned by the old eigenvectors. This would be no approximation to Eq. (28) if one would include all eigenvectors, because the old and new eigenvectors span the same space. In practice, however, the number of calculated states, N_{s}, is much smaller (by almost an order of magnitude) than the matrix size, N_{pw}. If one would choose N_{s} equal to the number of occupied states in the solid, N_{occ} the new eigenvectors would not be improved at all, since the new eigenvectors, C^{new}, would simply be a linear combination of the old eigenvectors. Here, we take $N_{\text{s}} = 2N_{\text{occ}}$ which was found to be a good compromise between accuracy and numerical effort.

The old eigenvectors are now viewed as an unitary transformation

$$C^{\text{old}^\dagger} C^{\text{old}} = S. \tag{31}$$

In the case $S = E$ (no overlap, E unity matrix) Eq. (31) always holds. In the general case Eq. (31) is only valid, if $S^{\text{new}} \simeq S^{\text{old}}$. This aspect must be especially considered in the case of the LAPW method because the basis functions are recalculated in each iteration. This problem can be overcome by transforming the generalized eigenvalue problem to a regular one (i.e. by Cholesky decomposition). Here, we chose to treat the generalized problem with the Block–Davidson scheme and the regular problem using the Lanczos algorithm. The reason for this strategy is the following: The Lanczos algorithm has due to its simplicity a very low numerical cost and thus can compensate for the extra cost of the Cholesky decomposition. Treating the overlap matrix S explicitly would require to orthogonalize the sets $H^i B^{i-1}$ using S as a metric tensor which would ruin the numerical effort saved by not doing the decomposition.

The reduced eigenvalue problem is then given by

$$\widetilde{H} = \varepsilon \widetilde{S} \widetilde{C}^{\text{new}} \quad \text{with} \tag{32}$$

$$\widetilde{H} = C^{\text{old}^\dagger} H C^{\text{old}}, \tag{33}$$

$$\widetilde{S} = C^{\text{old}^\dagger} S C^{\text{old}} \quad \text{and} \tag{34}$$

$$\widetilde{C}^{\text{new}} = S C^{\text{old}^\dagger} C^{\text{new}}. \tag{35}$$

The process of iterating the solution of Eq. (32) consists of optimizing the N_s basis functions initially given by $\widetilde{C}^{\text{new}}$ by adding $N_s - N_{\text{occ}}$ linear independent vectors to this set. In the subsequent discussion the set $\widetilde{C}^{\text{new}}$ consisting of N_s basisvectors will be named B^0 and the set of N_s basisvectors added in iteration i, B^i. The actual iteration procedure then consists of using $\{B^0, \ldots, B^i\}$ in Eqs. (32)–(34), to construct B^{i+1} from $\{B^0, \ldots, B^i\}$ and turn back to Eqs. (32)–(34). At the end of the iteration process the eigenvectors are obtained from Eq. (35). Here, the set $\{B^0, \ldots, B^i\}$ is viewed at as a rectangular matrix of size $N_{\text{pw}} \times (i+1)N_s$.

4.4.4. Lanczos scheme

As already mentioned above, we now take $S = E$. The basic idea is to improve B^i by the N_s vectors obtained from calculating HB^{i-1} and orthogonalizing this set to the set B^{i-1} (e.g., by Graham–Schmidt orthogonalization). In fact this is one of the easiest ways to increase the basis set, because in practice HB^{i-1} had to be calculated already in Eq. (33). To our knowledge, a strict mathematical proof that the series $HB, H^2B, \ldots, H^nB$ should converge to the eigenvectors of H does not exist, but experience has shown that this approach is fairly stable and accurate.

4.4.5. Block–Davidson scheme

This scheme uses a more subtle way to expand the basis B. In iteration i one gets from Eqs. (32)–(35) a current approximation to the true eigenvector $|c_j\rangle$, denoted as $|c_j^i\rangle$. The aim is to find a correction vector $|\delta A\rangle$ such that

$$|c_j\rangle = |c_j^i\rangle + |\delta A\rangle. \tag{36}$$

This correction vector $|\delta A\rangle$ can be formally calculated by plugging Eq. (36) into Eq. (28).

$$(H - \varepsilon_j S)|c_j^i\rangle = (H - \varepsilon_j S)|\delta A_j\rangle \tag{37}$$

The left side of Eq. (37) is called residual vector, $|R_j\rangle$. In principle the inversion of $(H - \varepsilon S)$ would yield the correct $|\delta A\rangle$, but in practice this is never done because the computational cost of this inversion would already be comparable to an exact diagonalization. Thus, one only retains the diagonal elements of $(H - \varepsilon S)$ to make the inversion trivial. Eq. (37) is then expressed with help of the basis B^i

$$|\delta A_j\rangle = \sum_k \frac{\langle b_k^i | R_j\rangle}{\langle b_k^i | H - \varepsilon_j S | b_k^i\rangle} |b_k^i\rangle. \tag{38}$$

The matrix containing the $|\delta A_1\rangle, \ldots, |\delta A_{N_s}\rangle$ is then used to increase the basis B^i to B^{i+1}.

5. Examples

In the following we demonstrate the effect of our improvements on a huge example, namely a nine layer slab of (4×2)-Cu(110) (i.e. 72 atoms, 792 valence electrons). We will compare the CPU-time needed to reach selfconsistency using our improved code with the original code. Additionally we will compare our LAPW code with a most efficient implementation of the PPW-method [25]. The Cu(110) surface is modeled by a nine layer slab repeated periodically in all three dimensions and separated by a vacuum zone equivalent to five substrate layers. We use a lattice constant of 6.64 bohr, which corresponds to the theoretical LDA bulk value. Since both methods scale almost linearly with the number of **k**-points, only one point in the surface BZ has been used for these benchmarks. The MT radii are chosen to be 2.20 bohr. The kinetic-energy cutoff for the plane wave basis needed for the interstitial region is set to 13.22 Ry which leads to matrix-sizes of the Hamiltonian matrix of about 7000×7000. The partial wave (l, m) representation (inside the MTs) is taken up to $l_{\max} = 10$. A plane-wave cutoff energy of 81 Ry for the Fourier representation of the potential is used. The maximum angular momentum in the (L, M) expansion of the potential inside the atomic spheres is set to $L_{\max} = 4$. In the PPW calculations, plane waves up to a kinetic energy of 70 Ry had to be used, to reach a comparable level of accuracy, but we also include the CPU-time required for a PPW calculation at 40 Ry.

Table 1
Distribution of CPU-time needed for the different parts of the generation of the new electron density (`lapw2`) comparing the original version ("WIEN95") with the new one ("optimized"). The column ("speed-up factor") lists the speed-up reached

	WIEN95		Optimized		Speed-up factor
	CPU-time	%	CPU-time	%	
Spheres	2 h 24 m	44	12 m 20 s	69	12
Interstitial	3 h 4 m	56	5 m 22 s	31	34
Total	5 h 28 m		17 m 42 s		18.5

5.1. LAPW2: *Generating the electron density*

The original code WIEN95 needed 19680 CPU-seconds (5 h 28 m) for the generation of the electron density on an IBM RS/6000 node (Table 1). The calculation of the electron density inside the spheres took 8640 CPU-seconds (2 h 24 m) (44%), while 11040 CPU-seconds (3 h 4 m) (56%) were needed for the interstitial electron density. On this latter part our improvements led to a reduction of the necessary CPU-time to 322 CPU-seconds (5 m 22 s), which is equivalent to a speed-up factor of 34. In the part generating the electron density inside the MT-spheres, the improvement is not as big, but still a speed-up factor of 12 could be reached, reducing the CPU-time to 740 seconds (12 m). The relative weight of the two tasks is shifted by the optimization to 70% for the spheres and 30% for the interstitial. With these improvement, the contribution of LAPW2 to the overall runtime becomes negligible, and thus all further considerations should focus on the program LAPW1 and its most time consuming part, the diagonalization of the Hamilton-Matrix.

5.2. LAPW1

As a general result it was found that in order to obtain reasonable accuracy in total energies it is sufficient for both methods, the Block–Davidson as well as the Lanczos scheme, to improve the expansion set only once, i.e. using $\{B^0, B^1\}$ to construct the new eigenvectors. Both iterative schemes worked well for the Cu(110) benchmark system. The speed-up gained with respect to the full diagonalization was 1.45 in the case of the Lanczos scheme and 3.12 in the case of the Block–Davidson scheme (Table 2). Fig. 3 illustrates the accuracy of both methods during the SCF-cycle. In the upper panel the overall performance is illustrated: The left panel shows the deviation of the total energies obtained by both methods with respect to the exact diagonalization result. Here, the largest deviation is about 1 mRy, but when self-consistency is approached, the deviations are well below the convergence criterion of 0.5 mRy. The right panel shows in an analogous way the deviations of the electron differences (the mean square deviation of $n_{\text{old}} - n_{\text{new}}$ inside the MT's) during the SCF-cycle. This gives an idea about the overall quality of the approximated eigenvectors. The deviation in the total energies are less than 0.3 mRy and thus show no essential difference between the two schemes, but the electron difference indicates that the Block–Davidson method leads to better eigenvectors especially during the first four cycles of the SCF-cycle. The quality of the eigenvectors is illustrated in more detail in the lower panel of Fig. 3 for the valence electron densities only. In the left (right) panel the mean square deviation between $n_{\text{exact}} - n_{\text{iter}}$ is evaluated inside the MT's (interstitial region), where "iter" stands for either the Davidson or the Lanczos method. It can be clearly seen that the Davidson method leads to results that are closer to the exact solution than the results obtained by the Lanczos-method, but again, when self-consistency is reached, both methods give essentially the same results for the interstitial region as well as inside the MT.

5.3. Total speed-ups

Table 3 shows the distributions of CPU-time needed for the different parts of the LAPW-self-consistency cycle for the original WIEN95 program as well as for our new, optimized code. The enormous speed-up factor close to 20

Table 2
CPU-time in LAPW1 needed for the setup (spherical and non-spherical H and S matrix) and the diagonalization; for the original code the standard diagonalization is used ("WIEN95"), while for the "optimized" version the timing for both, the Lanczos ("Lan") and the Block–Davidson ("Dav") method are given and the non-spherical part of the Hamiltonian (which is ignored for the first half of the iterations towards self-consistency) is the average over all iterations. The last column lists the corresponding speed-up factors

	WIEN95	Optimized	Speed-up factor
	CPU-time	CPU-time	
Spherical (H, S)	43 m 17 s	16 m 29 s	2.62
Non-spherical (H)	37 m 13 s	18 m 36 s	2.00
Diagonalization	1 h 12 m 5 s	Lan: 49 m 42 s	1.45
Diagonalization		Dav: 23 m 07 s	3.12
Total	2 h 32 m 35 s	Lan: 1 h 24 m 47 s	1.80
		Dav: 57 m 12 s	2.67

Table 3
Distribution of CPU-time needed for the different parts of the LAPW-self-consistency cycle comparing the original version ("WIEN95") with the new one ("optimized"). The last column shows the speed-up factor reached

	WIEN95		Optimized		Speed-up factor
	CPU-time	%	CPU-time	%	
lapw0	26 m	5	26 m	22	1.00
lapw1	2 h 33 m	30	57 m	61	2.67
lapw2	5 h 28 m	65	17 m	15	19.29
core	4 s		4 s		
mixer	2 m		2 m	2	
Total	8 h 29 m		1 h 46 m		4.80

for the program `LAPW2` indicates, that the original code, which was tuned for a vector machine, did not match the needs of modern high performance workstations with fast but small cache and relatively slow main-memory access. This extraordinary speed-up could not be gained for the program `LAPW1`. However, with the implementation of the new iterative diagonalization algorithms and the omission of the non-spherical terms to the Hamilton-matrix in the first half of the self-consistency run, the required CPU-time is cut down by a factor of 2.67. In total all our modifications lead to a speed-up of 4.80.

5.4. Comparing the computationally costs of FP-LAPW and pseudopotentials plane waves (PPW) codes

In order to compare our improved FP-LAPW-code with the PPW-approach, we also calculated our test system with the highly optimized PPW code `fhi96md` [25].

In this implementation of the PPW method, the Kohn–Sham-equations are solved by an iterative optimization of a set of trial wave functions, combining self-consistency and iterative matrix diagonalization, where the iteration of the wave functions is formulated in terms of equations of motion, as proposed by Car and Parinello [33]. In the `fhi96md` code a second order equation is used, which had been suggested by Joannopoulos [34]. In this scheme, each single step is computationally much cheaper than a single self-consistency cycle within the FP-LAPW-method, but the number of iterations needed to reach self-consistency is usually much larger and crucially

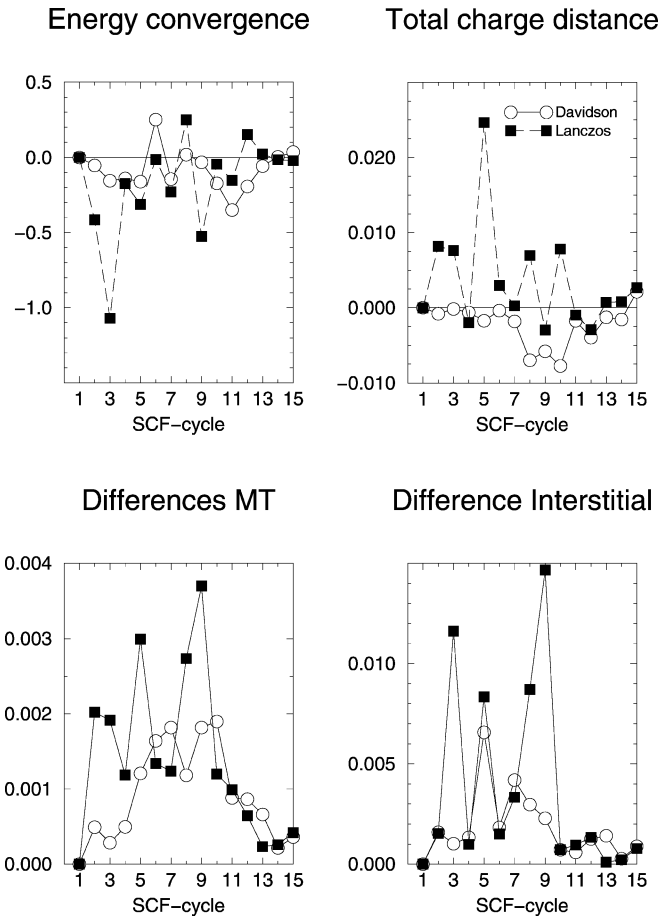

Fig. 3. Upper panel: Deviations of total energy (left panel) in mRy and electron distance (right panel) with respect to the exact diagonalization for the Lanczos (filled boxes) and Block–Davidson (open circles) method during the SCF-cycle. Lower panel: Valence electron distance to the exact solution (see text) for the MT-contributions (left panel) and plane wave contributions in interstitial region (right panel). At the start of the SCF-cycle an exact diagonalization is performed (no deviation) to obtain the input wavefunctions for the Lanczos- and Davidson-method, respectively.

depends on the quality of the initial guess for the wave functions. For this reason the `fhi96md` code employs a *mixed-basis-set* initialization, which gives starting wave functions of high quality. For details see Ref. [25].

Table 4 shows the CPU-time needed to converge our test system using both iterative matrix diagonalization methods. The initialization in the FP-LAPW-method is just the time needed to construct a starting electron density, whereas for the PPW-method the time reflects the set up of the starting wave functions within the mixed-basis scheme. As already mentioned, the time needed for a single iteration is much smaller for the `fhi96md` implementation of the PPW-approach than in the FP-LAPW-code, but this advantage is destroyed by the fact, that about five times as many iterations are needed to reach self-consistency. It should be noted that the meaning of "iteration" is in fact different in the FP-LAPW and the PPW method, as the PPW method [25] combines the iterative diagonalization with the selfconsistent update of the electron density. While the original `WIEN` code needed about 30% more CPU-time than the PPW-code to converge this system, our improved version is about three times faster.

It is important to note that Table 4 summarizes benchmark-calculations performed in summer 1997, and that the system Cu(110) with a (4×2) surface structure and 72 atoms per supercell was most favorable to identify

Table 4
CPU-time needed on an IBM RS/6000 node to converge a nine layers slab, representing a 4×2 Cu(110) surface cell. Comparison of the original WIEN95 code ("original"), our improved code ("optimized") and the fhi96md pseudopotential plane wave program ("PPW") with two different plane wave cutoffs ("70 Ry" and "40 Ry")

	FP-LAPW		PPW	
	Original	Optimized	70 Ry	40 Ry
$T_{\text{initialization}}$	30 m	30 m	18 h 40 m	8 h 45 m
$T_{\text{iteration}}$	8 h 24 m	1 h 46 m	1 h 7 m	30 m
$\#_{\text{iterations}}$	20	20	100	100
T_{total}	168 h 34 m	35 h 50 m	130 h 20 m	58 h 45 m

the advantages of the new FP-LAPW code, and at the same time it was least favorable for the plane-wave pseudopotential code fhi96md. In the meantime several improvements are being introduced in the pseudopotential code, as, for example, a real-space projector method [35] to evaluate the pseudopotential matrix-elements (which brings a speed up between a factor of 2 and 3), and ultra-soft pseudopotentials [36] (which brings a speed up by another factor of 2). Altogether, for the chosen benchmark system the new version of the plane-wave pseudopotential code, fhi99md, is about a factor of 20 faster, without loss in accuracy [37]. But we also note that for other systems the difference in CPU-time required for the new, fhi99md, and the older, fhi96md, code is much less pronounced.

Other plane-wave pseudopotential codes [38,39] also employ the mentioned improvements and behave similar to the fhi99md code. This discussion shows that comparisons between different methods (e.g., FP-LAPW versus plane-wave pseudopotentials) is indeed helpful to identify and optimize time critical algorithms and routines. With ever increasing system size program developments are getting more and more important. Although FP-LAPW was ahead the pseudopotential code (with respect to lower CPU-time consumption) for some systems in 1997 and 1998, recent improvements by introducing new concepts at the plane-wave pseudopotentials front make this again a more efficient code. We are convinced, however, that new concepts and techniques will also bring a speed up to FP-LAPW. Clearly FP-LAPW remains the most accurate tool and does not suffer from problems as linearization of core-valence exchange-correlation (which can be *partially* corrected in pseudopotential calculations), or the lack of core polarization (which may be important, e.g., for some magnetic systems). However, besides accuracy low CPU-time requirements are clearly very important. A fast (i.e. efficiently) working electronic structure code is crucial for present days problems, in particular to be able to test all relevant numerical approximations with the required care. We note that in many density-functional theory calculations performed for low symmetry and/or many-atom systems the main approximations are (often) not at the level of exchange-correlation functional but at the level of numerical approximations.

Although our test system may be a special case and other systems or a different computer architecture may lead to slight modifications, the fair estimate of the relative speed between FP-LAPW and PPW should remain valid.

6. Summary

The present work demonstrates that a continuous adaption of algorithms to the existing hardware architecture is indeed very important for efficient and accurate electronic structure calculations of many-atom systems. While the WIEN95 implementation of the FP-LAPW-method was optimized for a vector computer and performs well on those platforms, it is not well suited for modern cache-based processors. Our improvements led to a significant speed-up on those hardware achitectures and makes the FP-LAPW method a strong competition to the popular PPW approach. Especially for transition metal systems, the FP-LAPW method has a significant advantage. In

addition the FP-LAPW method gives as an all-electron method additional information about the system, which is out of reach for any pseudopotential method because of the frozen core approximation.

The significant improvements discussed here have been implemented in the new version WIEN97 of the FP-LAPW code [40] and the successful strategy adopted here may be useful for other software developers too.

Acknowledgements

We thank R. Reuter of IBM Heidelberg, Germany for his very helpful introduction into the art of producing high performance computer codes for numerically intensive computations. Helpful discussion with P. Kaeckell, who performed some calculations with the VASP [38] code, are greatfully acknowledged. P.B. and K.S. were supported in part by the Austrian Science Foundation (SFB project F1108), M.P. by the Deutsche Forschungsgesellschaft, Sonderforschungsbereich 290. This work was also supported by the TMR network "Electronic Structure Calculations of Materials Properties and Processes for Industry and Basic Sciences".

References

[1] J.C. Slater, Phys. Rev. 51 (1937) 846.
[2] J.C. Slater, Adv. Quantum Chemistry 1 (1964) 35.
[3] J. Bormet, J. Neugebauer, M. Scheffler, Phys. Rev. B 49 (1994) 17242.
[4] T.L. Loucks, Augmented Plane Wave Method (Benjamin, New York, 1967).
[5] L.F. Mattheiss, J.H. Wood, A.C. Switendick, Meth. Comp. Phys. 8 (1968) 64.
[6] J.O. Dimmock, Solid State Phys. 26 (1971) 103.
[7] H. Bross, Phys. Kondens. Mater. 3 (1964) 119; Z. Phys. B 81 (1990) 233.
[8] P. Marcus, Int. J. Quantum. Chem. Suppl. 1 (1967) 567.
[9] D.D. Koelling, J. Phys. Chem. Solids 33 (1972) 1335;
D.D. Koelling, G.O. Arbman, J. Phys. F 5 (1975) 2041.
[10] O.K. Andersen, Solid State Commun. 13 (1973) 133; Phys. Rev. B 12 (1975) 3060.
[11] E. Wimmer, H. Krakauer, M. Weinert, A.J. Freman, Phys. Rev. B 24 (1981) 864.
[12] H.J.F. Jansen, A.J. Freeman, Phys. Rev. B 30 (1984) 561.
[13] L.F. Mattheiss, D.R. Hamann, Phys. Rev. B 33 (1986) 823.
[14] P. Blaha, K. Schwarz, P. Sorantin, S.B. Trickey, Comput. Phys. Commun. 59 (1990) 399.
[15] D.J. Singh, Planewaves, Pseudopotentials and the LAPW Method (Kluwer Academic, Boston, 1994).
[16] P. Hohenberg, W. Kohn, Phys. Rev. B 136 (1964) 864.
[17] W. Kohn, L.J. Sham, Phys. Rev. A 140 (1965) 1133.
[18] P. Blaha, K. Schwarz, P. Herzig, Phys. Rev. Lett. 54 (1985) 1192.
[19] P. Dufek, P. Blaha, K. Schwarz, Phys. Rev. Lett. 75 (1995) 3545.
[20] K. Schwarz, C. Ambrosch-Draxl, P. Blaha, Phys. Rev. B 42 (1990) 2051.
[21] B. Winkler, P. Blaha, K. Schwarz, Am. Mineralogist 81 (1996) 545.
[22] B. Kohler, P. Ruggerone, S. Wilke, M. Scheffler, Phys. Rev. Lett. 74 (1995) 1387; in: Electronic Surface and Interface States on Metallic Systems, E. Bertel, M. Donath (Eds.) (World Scientific, Singapore, 1995).
[23] X.-G. Wang, W. Weiss, Sh.K. Shaikhutdinov, M. Ritter, M. Petersen, F. Wagner, R. Schlgl, M. Scheffler, Phys. Rev. Lett. 81 (1998) 1038.
[24] B. Kohler, S. Wilke, M. Scheffler, R. Kouba, C. Ambrosch-Draxl, Comp. Phys. Commun. 94 (1996) 31.
[25] M. Bockstedte, A. Kley, J. Neugebauer, M. Scheffler, Comp. Phys. Commun. 107 (1997) 187; http://www.fhi-berlin.mpg.de/th/fhimd/.
[26] P. Blaha, K. Schwarz, P. Dufek, R. Augustyn, `WIEN95` (Technical University, Vienna, 1995); improved and updated UNIX version of the original copyrighted `WIEN`-code [14].
[27] C. Lawson, R. Hanson, D. Kincaid, F. Krogh, ACM Trans. Math. Softw. 12 (1969) 308.
[28] C. Lawson, R. Hanson, D. Kincaid, F. Krogh, ACM Trans. Math. Softw. 12 (1969) 324.
[29] C. Lawson, R. Hanson, D. Kincaid, F. Krogh, ACM Trans. Math. Softw. 16 (1990) 1.
[30] http://www.rs6000.ibm.com/resource/technology/MASS/index.html.
[31] D. Kvasnicka et al., to be published;
E.J. Haunschmid, C.W. Ueberhuber, Hierarchically blocked cholesky factorization, in: Proceedings of the Second IASTED International Conference, European Parallel and Distributed Systems (Euro-PDS'98), pp. 335–343.

[32] For a review on iterative methods see, e.g., D.M. Wood, A. Zunger, J. Phys. A 18 (1985) 1343;
G. Kresse, J. Furthmüller, Phys. Rev. B 54 (1996) 11169;
G. Kresse, J. Furthmüller, Comp. Mat. Sci. 6 (1996) 15.
[33] R. Car, M. Parinello, Phys. Rev. Lett. 55 (1985) 2471.
[34] M.C. Payne et al., Phys. Rev. Lett. 56 (1986) 2656.
[35] R.D. King-Smith, M.C. Payne, J.S. Lin, Phys. Rev. B 44 (1991) 13063.
[36] D. Vanderbilt, Phys. Rev. B 41 (1990) 7892.
[37] J. Neugebauer, private communication.
[38] G. Kresse, J. Furthmüller, Phys. Rev. B 54 (1996) 11169.
[39] M.C. Payne, M.P. Teter, D.C. Allan, T.A. Arias, J.D. Joannopoulos, Rev. Mod. Phys. 64 (1992) 1045.
[40] P. Blaha, K. Schwarz, J. Luitz, WIEN97, A Full Potential Linearized Plane Wave Package for Calculating Cristal Properties, K. Schwarz, Techn. Universität Wien, Austria, 1999. ISBN 3-9501031-0-4.