ELSEVIER

Computational Materials Science 25 (2002) 478–492

COMPUTATIONAL MATERIALS SCIENCE

www.elsevier.com/locate/commatsci

First-principles computation of material properties: the ABINIT software project

X. Gonze [a,*], J.-M. Beuken [a], R. Caracas [a], F. Detraux [a], M. Fuchs [a], G.-M. Rignanese [a], L. Sindic [a], M. Verstraete [a], G. Zerah [b], F. Jollet [b], M. Torrent [b], A. Roy [b], M. Mikami [c], Ph. Ghosez [d], J.-Y. Raty [d], D.C. Allan [e]

[a] *Unité PCPM, Université Catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium*
[b] *Commissariat à l'énergie atomique, F-91680 Bruyères-le-Chatel, France*
[c] *Mitsubishi Chemical Corp., 1000 Kamoshida-cho Aoba-Ku, Yokohama, 227-8502 Japan*
[d] *Département de Physique, U. de Liège, B-4000 Sart-Tilman, Belgium*
[e] *Corning Inc., Corning, NY 14830 USA*

Received in revised form 15 February 2002; accepted 3 May 2002

Abstract

The density functional theory (DFT) computation of electronic structure, total energy and other properties of materials, is a field in constant progress. In order to stay at the forefront of knowledge, a DFT software project can benefit enormously from widespread collaboration, if handled properly. Also, modern software engineering concepts can considerably ease its development. The ABINIT project relies upon these ideas: freedom of sources, reliability, portability, and self-documentation are emphasised, in the development of a sophisticated plane-wave pseudopotential code.

We describe ABINITv3.0, distributed under the GNU General Public License. The list of ABINITv3.0 capabilities is presented, as well as the different software techniques that have been used until now: PERL scripts and CPP directives treat a unique set of FORTRAN90 source files to generate sequential (or parallel) object code for many different platforms; more than 200 automated tests secure existing capabilities; strict coding rules are followed; the documentation is extensive, including online help files, tutorials, and HTML-formatted sources.
© 2002 Elsevier Science B.V. All rights reserved.

PACS: 71.15.-m; 81.05.Zx; 31.15.Ew; 89.80.+h
Keywords: Density functional theory; Software engineering; Electronic structure

1. Introduction

The first-principles computation of material properties, relying upon quantum mechanics and electromagnetism, has undergone tremendous progress in the past twenty years. The widespread density functional theory (DFT) [1,2] is at the heart of this rapid evolution. It has been implemented in different computer codes, as well as generalised (e.g. time-dependent DFT [3]), or used as a basis for more sophisticated formalisms.

* Corresponding author.

Without being comprehensive, let us mention a few of the milestones in this field since the mid-eighties, in view of showing its continuous growth, with a special focus on the use of pseudopotentials.

In 1985, Car and Parrinello proposed an algorithm [4] that unified DFT with molecular dynamics. Independently, the same year, the so-called "GW" approximation of Hedin [5] for computing one-electron addition or removal energies, was implemented by Hybertsen and Louie [6]. In 1987, the linear-response approach to the dynamical and dielectric properties of solids was implemented by Baroni et al. [7], while Allan and Teter [8] proposed to use separable (Kleinman–Bylander) [9] pseudopotentials in conjunction with the Car–Parrinello technique. Vanderbilt designed ultrasoft pseudopotentials in 1990 [10], and three years later, together with King-Smith, identified the electronic polarisation as a geometric phase [11]. At about the same time, implementations on massively parallel computers appeared [12]. Further elaboration on the pseudopotential concept lead to Blöchl's projector augmented waves method [13]. In order to tackle excited states, both time-dependent DTF [14,15] and the Bethe–Salpeter equation formalism [16–18] were recently implemented.

Thus, in order to stay up-to-date, computer programs for first-principles study of materials must include more and more functionalities, that progressively become considered as "basic" functionalities. It is obviously not efficient anymore to test a new idea on the basis of a computer code that would have been developed from scratch for that purpose. Furthermore, a single individual cannot keep up with the maintenance and development of a software in which more and more functionalities must be incorporated, without sacrificing his own research. It is also clear that improvements or generalisations of the DFT formalism, and/or its implementation, and/or its domain of application, will continue for the next decade or even longer. Group development is required, and even international collaboration. In this context, different modern software engineering techniques prove extremely useful.

The ABINIT software project was started in 1997, on this basis, as an open software project, without a "definite" goal, developed using several software engineering techniques to allow international collaboration of many different groups. The main program of the ABINIT package performs density functional calculations of material properties, using a plane-wave basis and pseudopotentials. Special functionalities deal efficiently with response functions.

At present, this software is (1) open source (available under the GNU General Public License [19]), (2) self-testing, (3) portable across platforms for serial and parallel execution, (4) self-documented. A self-learning procedure is provided to the user. A protocol for international group development has been set up, including an explicitly stated coding style. The web site [20] provides the official versions, pseudopotentials, different utilities, benchmarking results, mailing lists and bibliographical information.

In the present paper, we refer to the specific version 3.0 of ABINIT. Previous versions of the package were not provided under GNU GPL, and the web site access was restricted.

The main program in ABINIT, as well as some utilities, are written in FORTRAN90 (more than 300 routines, about 100 000 lines, roughly a third of which are comment lines). The package also includes documentation files, scripts for automatic testing, pseudopotential files.

ABINITv3.0 was released first in December 2000. Since then, bug fixes have been regularly reported and posted on the web (currently version 3.0.6). On the other hand, the mainstream of development is continuing beyond version 3.0, with the existing versions 3.1 to 3.4. The present authors of this paper have made major contributions to ABINITv3.0, although more than 50 individuals have contributed to it, usually through modifications of one or a small number of routines, or through bug fixes.

Many other codes based on plane waves and pseudopotentials or projector-augmented waves codes (the latter considered as an improvement of the ultrasoft pseudopotential formalism) are available today. The list that we provide here [21–31], is likely not exhaustive. Some of them are commercialised, some others are available through collaboration with the main development team, or available without restriction from the web.

In the future, it might be interesting to develop some collaboration with the development teams that provide their code under the GNU General Public License, (that provide legal protection of the rights of both developers and users [21–23]), or the teams that provide their code freely [24–26]. Collaborative work might involve cross-checking of accuracy and speed, as well as sharing of routines or libraries. Obviously, open source development will be essential in this respect.

In this article, we will first focus on the capabilities of ABINITv3.0 (Section 2). Some of them are basic features, which have been used in similar codes for a long time, and will not be explained in much detail. Others are features of which we know no other implementation, and will be the subject of a longer explanation, although a complete description should be published elsewhere. We will also briefly present the structure of the package and the organisation of input files. Then, in Section 3, we will describe the organisation of the worldwide group development, made possible thanks to modern software engineering concepts: self-testing, scripts for portability, self-documentation, user self-learning. In the final Section 4, we will critically discuss the present achievements and the development model that we have followed, and indicate possible improvements.

2. Features

The ABINIT package includes different programs: the main ABINIT program, and the utilities MERGE, IFC and CUT3D. The main ABINIT program is a driver for different density functional based calculations. In what follows, we will distinguish (i) electronic ground-state capabilities, (ii) structure-related capabilities (optimisation and molecular dynamics), and (iii) response-function capabilities. References to the adequate publications are provided, with only a brief description, for most of the features. However, in two cases, namely, the treatment of the spin–orbit (SO) interaction and the adiabatic-connection fluctuation dissipation theorem (ACFDT), the ABINIT implementation is state-of-the-art, and more explanation will be given. The utilities MERGE, IFC and CUT3D will be described in a separate section.

2.1. Electronic ground-state capabilities

The pseudopotential plane-wave technique for density functional calculations has been reviewed by Payne et al. [33].

In ABINITv3.0, the following density functional approximations are supported: the local (spin-)density approximations of Perdew–Zunger [34], Teter and co-workers [35], the generalised gradient approximation (GGA) of Perdew et al. [36], the GGA potentials of van Leeuwen–Baerends [37], as well as the older non-spin-polarised local density approximations (LDA) of Gunnarsson–Lundqvist [38], Wigner [39], and Slater (X-alpha) [40].

The pseudopotentials that ABINIT is able to use, in separable form [9], are of many different types:

- standard norm-conserving pseudopotentials on a numerical grid, such as (but not limited to) Troullier–Martins pseudopotentials [41];
- Goedecker et al. [35] or Hartwigsen et al. [42] pseudopotentials, whose analytical form is particularly compact;
- extended norm-conserving pseudopotentials, as proposed by Teter [43].

On the web site, two complete (or nearly complete) sets of LDA pseudopotentials are provided: one of the Troullier–Martins type, and one of the Hartwigsen–Goedecker–Hutter type. The Fritz–Haber-Institute code [44] can be used to generate new pseudopotentials, in particular those to be used within the GGA. The Hartwigsen–Goedecker–Hutter pseudopotentials have a spin–orbital part, while the latest version of the Martins pseudopotential code [45] can also be used to generate SO dependent pseudopotentials. Non-linear exchange-correlation core corrections, as proposed by Louie et al. [46] or Teter [43] can be treated. Finally, core-hole pseudopotentials, needed to study core-level chemical shifts [47] can also be used.

The wavefunctions at each k-point (wavevector in the reciprocal space, usually in the first Brillouin Zone) are represented by the numerical coefficients of a finite set of plane waves, determined by a kinetic energy cut-off. The k-point sets can be generated automatically, following the Monkhorst–Pack scheme [48], or a generalisation thereof. The production of electronic band structure is made easy thanks to the possibility of automatically generating k-points that are regularly spaced along different lines in reciprocal space.

Symmetries are used to decrease the number of k-points needed to sample the Brillouin zone, so that only the irreducible part of it must be sampled.

Different possibilities are offered for the wavefunction representation in reciprocal space (plane-wave coefficients). In the most frequent case, the wavefunction coefficients are complex valued wavefunctions, scalar in spinor space. However, ABINITv3.0 can treat collinear magnetism (ferromagnetism and anti-ferromagnetism), using independent spin up and spin down wavefunctions, as well as the SO coupling, using spinor wavefunctions. The ability to treat SO coupling is a rather advanced feature of ABINIT, and will be described in a separate section. In ABINITv3.0, the ability to treat non-collinear magnetism is still missing. For the specific wavevectors invariant under time-reversal (and translation by a vector of the reciprocal lattice), e.g. (0 0 0) or (1/2 0 0) . . ., the wavefunction is represented on the adequate halved set of plane-wave coefficients.

The algorithm that determines the wavefunctions, in a trial effective potential, is an adaptation of the conjugate-gradient algorithm of Teter et al. [49]. The density is constructed from these wavefunctions, and processed to find the Hartree and exchange-correlation part of the potential. In this way, an input trial potential is associated to an output potential. Different algorithms allow to iteratively build trial potentials that converge towards the Kohn–Sham potential (fixed point where the input and output potentials are identical): simple mixing [50], the Anderson algorithm [51], and the potential-based conjugate gradient algorithm [52,53]. Preconditioning of these algorithms is achieved through a model dielectric function, or through an approximate dielectric matrix [54].

When the density has been built self-consistently, the corresponding potential can be used non-self-consistently, to generate unoccupied states (needed for the generation of band structure or for the ACFD formalism, see Section 2.5).

The treatment of state-dependent occupation numbers can be done in a number of ways: these can be set up by hand (collectively for each band, or individually for each spin, k-point or band number), or filled automatically. In the latter case, the user can choose between semiconductor filling, or metallic filling, according to different smearing schemes: gaussian smearing (Fu and Ho [55]), Hermite–Gauss smearing (Methfessel and Paxton [56]), cold smearing (Marzari [57]).

2.2. Structure-related capabilities

Different choices of symmetry specification are provided. Given the atomic positions in the unit cell, the symmetry operations that leave them invariant are automatically recognised. Conversely, given the symmetry operations, and the list of irreducible atoms, the remaining atoms in the unit cell can be automatically generated.

It is also possible to input the number of the space group, according to the international tables for crystallographic [58], in which case the list of symmetry operations is initialised thanks to a database. Given the lattice parameters and symmetry operations, ABINIT is able to find the Bravais lattice and the point symmetry group (not yet the space group in ABINITv3.0).

The computation of forces and stresses, thanks to the Hellmann–Feynman theorem or the stress theorem within DFT [59,60], is implemented in all the cases in which the total energy can be computed, except in the SO case (i.e., it is available for scalar wavefunctions as well as spin-polarised ones, metallic as well as insulating materials, and for the different exchange-correlation functionals).

Forces and stresses can be used either to generate an optimised structure (minimising the forces and stresses, optionally constraining some of their components), using the Broyden algorithm [61], the modified Broyden algorithm [62], the Verlet algorithm with sudden atomic stop when the ion moves against the force [63], or to generate molecular dynamics trajectories, using the Verlet algorithm [64] or Numerov algorithm [65]. In the case of Verlet algorithm, it is possible to sample the canonical ensemble using the Nosé–Hoover and Langevin thermostat [66–70].

The code provides an automatic analysis of bond lengths and angles, as well as the atomic coordinates in *xyz* format.

2.3. Response-function capabilities

We first consider responses to atomic displacements and static homogeneous electric fields.

The dielectric polarisation can be computed within the Berry phase formulation [11]. This feature is available for insulators, magnetic or non-magnetic, but not yet when SO splitting is present.

The linear-response technique (or density functional perturbation theory) [7,71–74] can be used to compute responses to atomic displacements and homogeneous electric fields. This produces dynamical matrices at selected wavevectors, the Born effective charges, or the (electronic) dielectric constant. These quantities are gathered in a database, that can be analysed by the MERGE and IFC codes (see later). In ABINITv3.0, linear responses are not yet available for the spin-polarised case, or for spinor wavefunctions, or for the GGA to the exchange-correlation functional.

Susceptibility matrices and dielectric matrices (at zero frequency, and for reciprocal space lattice vectors) are computed thanks to the sum over states formulation [77,78].

Electronic excitations can also be investigated, within the time-dependent DFT [14,15].

2.4. Spin–orbit coupling

The SO coupling is not often available in plane-wave pseudopotential codes. Its implementation in ABINITv3.2, where forces, stresses and response functions are available, will be the subject of a detailed presentation [79]. We present here a brief description of the implementation of total energy calculations in ABINITv3.0.

Starting from the Dirac equation, it is possible to obtain a Schrödinger-like equation mixing the great and small component of the relativistic wavefunction, valid to the order $1/c^2$ (see [80]). A pseudopotential can then be generated for each channel $l+\frac{1}{2}$ and $l-\frac{1}{2}$ An adequate linear combination of these pseudopotentials leads to a scalar relativistic (SR) part v_i^{SR} and to a SO part v_l^{SO}. Following Refs. [42,81], we combine again these pseudopotentials, and write the electron–ion potential as the sum of a SR term and of a SO term such as:

$$V_{ei} = \sum_l V_l^{\mathrm{SR}}(\mathbf{r},\mathbf{r}')|ls\rangle\langle ls| + \sum_l V_l^{\mathrm{SO}}(\mathbf{r},\mathbf{r}')\mathbf{L}\cdot\mathbf{S}|ls\rangle\langle ls|. \tag{1}$$

Each term of the $V_l^{\mathrm{SR}}(\mathbf{r},\mathbf{r}')$ or $V_l^{\mathrm{SO}}(\mathbf{r},\mathbf{r}')$ kind is cast in a factored form of the Kleinman–Bylander type, the extension to more than one projector being straightforward.

The standard plane-wave basis is extended to the spinor plane-wave basis, whose elements are two-component plane waves denoted by $\langle\mathbf{G},\sigma|$. A generic matrix element takes the form:

$$\langle\mathbf{G},\sigma|V_l^{\mathrm{SR}}|l,s\rangle\langle l,s|\mathbf{G}',\sigma'\rangle \tag{2}$$

for the spin independent part, and

$$\langle\mathbf{G},\sigma|V_l^{\mathrm{SO}}\mathbf{L}\cdot\mathbf{S}|l,s\rangle\langle l,s|\mathbf{G}',\sigma'\rangle \tag{3}$$

for the SO part. The result for the spin-independent term, in the Kleinman–Bylander formulation, is well known:

$$\langle\mathbf{G},\sigma|V_l^{\mathrm{SR}}|l,s\rangle\langle l,s|\mathbf{G}',\sigma'\rangle = 4\pi(2l+1)f_l(|\mathbf{G}|)f_l(|\mathbf{G}'|)P_l(\widehat{\mathbf{G}}\cdot\widehat{\mathbf{G}}'), \tag{4}$$

where $\widehat{\mathbf{G}} = -\mathbf{G}/|\mathbf{G}|$, $f_l(\mathbf{G})$ is the Kleinman–Bylander form factor in the reciprocal space, and P_l the Legendre polynomials. This results from the addition theorem:

$$\langle\mathbf{G}|l,s\rangle\langle l,s|\mathbf{G}'\rangle = \sum_m \langle\mathbf{G}|Y_{lm}^*Y_{lm}|\mathbf{G}'\rangle = \frac{(2l+1)}{4\pi}P_l(\widehat{\mathbf{G}}\cdot\widehat{\mathbf{G}}'). \tag{5}$$

To achieve an analogous formula for the SO term, we first consider the vector $\mathbf{L}|\mathbf{G}'\rangle$. Using the definition, $\mathbf{L} = \mathbf{r} \times \mathbf{p}$, and the identities $\mathbf{r}|\mathbf{G}\rangle = -i\nabla_{\mathbf{G}}|\mathbf{G}\rangle$ and $\mathbf{p}|\mathbf{G}\rangle = \mathbf{G}|\mathbf{G}\rangle$, we obtain: $\mathbf{L}|\mathbf{G}'\rangle = -i\mathbf{G}' \times \nabla_{\mathbf{G}'}|\mathbf{G}'\rangle$. We therefore transform the matrix element into

$$-i\langle\sigma|\mathbf{S}|\sigma'\rangle \cdot \mathbf{G}' \times \nabla_{\mathbf{G}'}\langle\mathbf{G}|V_l^{\mathrm{SO}}|l,s\rangle\langle l,s|\mathbf{G}'\rangle. \tag{6}$$

The computation of the last term mimics the computation of the spin-independent term. Noting that the $\nabla_{\mathbf{G}'}|\mathbf{G}'|$ terms do not contribute, due to the cross product with $\mathbf{G}'$, we finally obtain:

$$\begin{aligned}\langle\mathbf{G},\sigma|V_l^{\mathrm{SO}}\mathbf{L}\cdot\mathbf{S}|l,s\rangle\langle l,s|\mathbf{G}',\sigma'\rangle \\ = -4\pi i(2l+1)\langle\sigma|\mathbf{S}|\sigma'\rangle \cdot \mathbf{G}\times\mathbf{G}'P_l' \\ \times(\widehat{\mathbf{G}}\cdot\widehat{\mathbf{G}}')f_l(|\mathbf{G}|)f_l(\mathbf{G}'|),\end{aligned} \tag{7}$$

where P_l' is the first derivative of P_l.

From these formulae, the total energy of the system can be computed, as well as its derivatives with respect to atomic displacements and unit cell deformations to obtain forces and stresses.

2.5. Adiabatic-connection fluctuation dissipation theorem

The LDA and GGA within DFT often yield a usefully accurate description of the physical and chemical behavior of solids, surfaces, and molecules. In particular, they can give a realistic account of the atomic structure as well as elastic and vibrational properties. Regarding molecular interactions and the related potential energy surfaces, GGA's typically improve over the LDA, but yet more accurate functionals are needed to overcome critical limitations: GGA's still fail to predict the energetics of chemical reactions (molecular dissociation energies, heats of reaction, and activation energy barriers) with "chemical accuracy", and, like the LDA, do not properly include van der Waals interactions between distant subsystems.

The ACFDT allows one to generate fully non-local exchange-correlation functionals that include, for instance, the van der Waals interaction or exact non-local exchange, and thus go beyond traditional local-density or gradient corrected approximations. Explicitly, the exchange-correlation energy for an electronic system with density $n(\mathbf{r})$ is given by [82]

$$\begin{aligned}E_{\mathrm{XC}}[n] = \frac{1}{2}\int_0^1 \mathrm{d}\lambda \int \mathrm{d}^3\mathbf{r}\,\mathrm{d}^3\mathbf{r}'\frac{e^2}{|\mathbf{r}-\mathbf{r}'|} \\ \times\left[-\frac{\hbar}{\pi}\int_0^\infty \mathrm{d}u\chi_\lambda(\mathbf{r},\mathbf{r}';iu) - n(\mathbf{r})\delta(\mathbf{r}-\mathbf{r}')\right],\end{aligned} \tag{8}$$

where $\chi_\lambda(\mathbf{r},\mathbf{r}';(iu))$ is the (imaginary-frequency) dynamical density response function of the system with the electrons interacting by a scaled Coulomb potential $\lambda e^2/|\mathbf{r}-\mathbf{r}'|$ and moving in a modified external potential such that the density stays the same as for the physical ($\lambda = 1$) groundstate. For $\lambda = 0$, one deals with the non-interacting Kohn–Sham system; its response function is given explicitly by the Kohn–Sham eigenstates $\phi_{k\sigma}(\mathbf{r})$ and eigenvalues $\varepsilon_{k\sigma}$ as

$$\begin{aligned}\chi_0(\mathbf{r},\mathbf{r}';iu) = \sum_{\sigma,k,l}\frac{(\gamma_{k\sigma}-\gamma_{l\sigma})}{i\hbar u-(\varepsilon_{l\sigma}-\varepsilon_{k\sigma})}\phi_{k\sigma}^*(\mathbf{r})\phi_{l\sigma}(\mathbf{r}) \\ \times\phi_{l\sigma}^*\mathbf{r}'\phi_{k\sigma}(\mathbf{r}'),\end{aligned} \tag{9}$$

where the sum includes all occupied ($\gamma_{k\sigma} = 1$) and unoccupied ($\gamma_{k\sigma} = 0$) states. For $\lambda > 0$, the interacting and the Kohn–Sham response functions are related by a Dyson-type screening equation

$$\begin{aligned}\chi_\lambda(\mathbf{r},\mathbf{r}';iu) = \chi_0(\mathbf{r},\mathbf{r}';iu) + \int \mathrm{d}^3\mathbf{r_1}\mathrm{d}^3\mathbf{r_2}\chi_0(\mathbf{r},\mathbf{r_1};iu) \\ \times f_\lambda^{\mathrm{HXC}}(\mathbf{r_1},\mathbf{r_2};iu)\chi_\lambda(\mathbf{r_2},\mathbf{r}';iu),\end{aligned} \tag{10}$$

where $f_\lambda^{\mathrm{HXC}}(\mathbf{r},\mathbf{r}';iu) = \lambda e^2/|\mathbf{r}-\mathbf{r}'| + f_\lambda^{\mathrm{XC}}(\mathbf{r},\mathbf{r}';iu)$ is the Coulomb and exchange-correlation kernel, established in the context of time-dependent DFT [83]. The correlation part of E_{XC} and the exchange part can be separated as shown in Ref. [84]. The set of Eqs. (8)–(10) may be referred to as ACFDT formalism. In principle, it yields the exact density functional exchange-correlation energy. In practice, different approximate functional can be generated from different approximations to the dynamical density response function: by using specific approximations for the (unknown) time-dependent exchange-correlation kernel, the Kohn–Sham response function, and/or the solution of the Dyson equation.

In ABINITv3.0 [85], the initial Kohn–Sham groundstate [and thus $\chi_0(\mathbf{r},\mathbf{r}';iu)$] is calculated within the LDA or the GGA. The response functions and kernels are considered at the Γ-point and treated in the plane-wave representation defined, e.g. for $\chi_{0\mathbf{GG}'}(iu)$, by $\chi_0(\mathbf{r},\mathbf{r}';iu) = \sum_{\mathbf{G},\mathbf{G}'}\chi_{0\mathbf{GG}'}(iu) \times \mathrm{e}^{\mathrm{i}\mathbf{Gr}}\mathrm{e}^{-\mathrm{i}\mathbf{G}'\mathbf{r}'}$, where $\mathbf{G}$ is a reciprocal lattice vector. We obtain the Kohn–Sham response function thanks to the sum over states in Eq. (9). Using different time-dependent exchange-correlation kernels [85] we then solve the Dyson Eq. (10) without further approximations as the system of linear equations

$$\sum_{\mathbf{G}_1,\mathbf{G}_2}\left\{\delta_{\mathbf{GG}_1\cdot} - \chi_{0\mathbf{GG}_2}(iu)f^{\mathrm{HXC}}_{\lambda\mathbf{G}_2\mathbf{G}_1}(iu)\right\}\chi_{\lambda\mathbf{G}_1\mathbf{G}'}(iu) = \chi_{0\mathbf{GG}'}(iu). \tag{11}$$

To obtain E_{XC} we evaluate the correlation energy as

$$E_{\mathrm{C}} = \int_0^1 \mathrm{d}\lambda\frac{\hbar}{2\pi}\int_0^\infty \mathrm{d}u\sum_{\mathbf{G}}\frac{4\pi e^2}{\mathbf{G}^2}\{\chi_{0\mathbf{GG}}(iu) - \chi_{\lambda\mathbf{GG}}(iu)\}. \tag{12}$$

The exchange energy functional is then added [86–88]. For the integration over λ and u, we repetitively solve the Eqs. (9) and (11) and use Gaussian quadrature formulas.

This implementation is still at an experimental stage: it needs optimisation. The primary goal of our research is to assess the accuracy of different functionals of the ACFDT family.

2.6. Parallelisation

We have implemented different levels of parallelisation: distribution of k-points on different processors, distribution of the work related to different states within a given k-point, distribution of the work related to different wavefunctions coefficients. There is still room for optimisation of these different levels, so that it will not be the purpose of this section to present a detailed analysis of the speed-up, only the main ideas behind the parallelisation.

The most efficient parallelisation (large amount of computation with respect to the communications) is based on the distribution of k-points. It is implemented using the MPI library, and is available in all cases (electronic structure, total energy determination, response functions) where there is more than one k-point. It works for massively parallel or SMP machines, as well as for clusters with network-based communications. Unfortunately, the scaling of the number of k-points with the size of the unit cell is unfavourable: it decreases with increasing number of atoms. Still, there are many cases in which both the workload and the number of k-points is large (e.g. for metals, or for the response-function properties of small cells). As an example, a typical speed-up of 20 can be attained with 25 processors on a SGI Origin 2000 (processors R12K, 300 MHz), while on a cluster of 25 Intel Pentium III under Linux (550 MHz), ethernet 100 Mbits/s, the speed-up is still above 15.

The spread of the work related to different states within a given k-point is implemented only for the case of the response functions in ABINITv3.0. As with the distribution of k-points, it is implemented using the MPI library, and works for massively parallel or SMP machines, as well as for clusters with network-based communications. Its scaling is linear with the size of the system, and counterbalances the unfavourable scaling of the k-point spread: the product of the number of k-points by the number of states is roughly constant, for similar systems treated with similar accuracy, and typically vary between 200 and more than 1000. The amount of communication between processors, for response-function calculations, is only marginally larger for this state-based parallelisation than for the k-point distribution. However, in ABINITv3.0, the memory need per processor is not decreased by adding more processors.

Finally, it is also possible to distribute the work related to the coefficients of the wavefunctions over the processors of an SMP machine, thanks to OpenMP compiler directives. The FFT algorithm has been parallelised by this technique, as well as the application of the non-local operator to the wavefunction. However, this implementation is the most recent of the three parallelisations, and is far from being efficient: a typical speed-up of 2 has been observed using four processors. Interestingly, the MPI and OpenMP parallelisation can be used at the same time.

2.7. Utilities

The MERGE and IFC utilities are used to analyse the database of derivatives of the total energy with respect to atomic displacements and homogeneous electric field perturbations, produced by the main ABINIT program. This database can contain dynamical matrices and Born effective charges, as well as the electronic dielectric tensor. MERGE is a database handling tool, while the analysis is performed with IFC.

From these data, following Ref. [74], IFC can build:

(1) the interatomic force constants, including their asymptotic behaviour, based on Born effective charge tensors and the electronic dielectric tensor;
(2) the dynamical matrices at any point in the Brillouin zone, by Fourier interpolation of the dynamical matrices provided in the database, and thus the corresponding eigenvectors and eigenvalues, the latter forming the phonon band structure;
(3) symmetry characters of the phonons at zero wavevector;
(4) thermodynamical properties (such as free energy, heat capacity and entropy), in the quasiharmonic approximation, obtained by the integration of the phonon degrees of freedom over the Brillouin zone, with Bose–Einstein occupation factors;
(5) the frequency-dependent dielectric tensor, for frequencies lower than the electronic gap.

The CUT3D utility performs the analysis of the real-space three-dimensional density or potential files provided by the main ABINIT program. In particular, it is able to interpolate the value of the density or potential known in a three-dimensional arbitrary cartesian volume, on a two-dimensional plane that cuts the three-dimensional cell, along a line, or at any single point. It is also able to reformat the density or potential file for use as input to graphical softwares like MATLAB [75] or MOLEKEL [76].

3. Organisation of the software development

As mentioned in the introduction, the ABINIT project relies upon a large number of developers, belonging to different international teams. Software engineering techniques have been developed by computer scientists in order to allow the harmonious development of such projects. Some of them involve auxiliary software (usually available under the GNU General Public License), others are mostly concepts that guide the development effort. Several of these techniques have been adopted in the ABINIT project, and will be described here. Some others are not yet used, and will be part of Section 4.

3.1. Sharing the software: GNU General Public License

The GNU organisation has put forward the important concept of "free software": the software is copyrighted by the developers, but distributed under a license that guarantees the right of users to have access to the sources, the right to modify them and even the right to redistribute them. This concept has been further elaborated to give legal texts (available on the GNU web site [19]). The so-called "GNU General Public License" [32] is used for the vast majority of free softwares available today.

We have chosen to deliver ABINIT under this license. Each source file or documentation file in the ABINIT package bears the copyright of its author(s), as well as the distribution license. Thanks to this approach, the developers retain the acknowledgement of their effort, while allowing others to improve their work in future releases. The user is guaranteed to have access to the source code, and can actually contribute to the debugging effort.

3.2. Automatic tests

Modifications performed by some developer might introduce bugs. This obvious source of problems in code development is further amplified when a large group is involved: most developers only know a restricted part of the code, while the level of care of the developers can vary widely. In order to avoid this drawback, self-testing has been implemented.

As an integral part of the ABINIT package, more than two hundred tests are delivered. The testing environment includes one input file per test, the pseudopotentials, and scripts for running batches of tests and analysing their results automatically. For each functionality of ABINIT, there exists one (or more) such test, created during the implementation. On average, each of these tests runs in a dozen seconds on a PC (Pentium III at 800 MHz).

The ultimate goal of the automatic testing environment is to provide to the developer a (one-bit) positive signal when everything is fine, and an adequate failure analysis (more than one bit, of course) otherwise. In ABINITv3.0, the analysis is multi-level:

(1) A first script, called “fldiff” (for “floating-diff”), written in PERL, is able to compute the difference between a resulting file and a reference file, in such a way that floating point inaccuracies, inherent to the variability of platforms and computers, are ignored to within some bound. This tool requires the resulting and reference files to be rather similar, except for the floating point differences. When the test is successful, only a few lines are written in a summary file, that gathers the results of the batch of tests done automatically.
(2) When a non-negligible difference is obtained, the result of a usual UNIX “diff” command is available automatically. This allows for easy examination of more complex differences.
(3) For each run of ABINIT, there is also a detailed “log” file, usually ignored in production runs, in which error, warning, and comment messages are delivered.

Every contributor to the ABINIT effort is required to set up a test when making his development effort. The modified source files, the input file and the reference file for the tests are included in the next version of ABINIT. The delivery of the reference file by the developer to the person in charge of the official version is essential, and allows one to insure that the new functionality is indeed safe.

Most of the tests are suited for the sequential and OpenMP versions of ABINIT. A separate set of tests focuses on the comparison of the sequential and MPI-parallel execution.

Finally, there is also a small class of tests that do not examine the functionalities of ABINIT, but benchmark the speed: they compare different versions of the most CPU-critical routines and allow the user to compare machine speeds.

3.3. Portability

ABINITv3.0 has been installed on the following platforms:

(1) PC, based on Pentium Pro, Pentium II, and Pentium III processors, under Linux, with the PGI compiler (Portland Group Inc.), as well as the Fujitsu compiler;
(2) PC, based on Intel 486 and Pentium II under Windows 98 or NT, using the PGI workstation suite;
(3) Compaq, based on alpha processors (EV56, EV6 or EV67), under OSF;
(4) Compaq, based on alpha processors (EV56), under LINUX;
(5) IBM RS6000, based on Power 2 and 3+ processors (model 590, 3CT, nighthawk);
(6) SGI Origin 2000, based on R1000 processors;
(7) CRAY T3E;
(8) FUJITSU VPP700;
(9) Sun UltraSparc II;
(10) NEC SX4 and SX5;
(11) HITACHI SR8000;
(12) Macintosh.

The optimisation of the main code is quite advanced for the platforms (1)–(7), and could be better for the other platforms. Binaries for most of these machines, contributed by different developers, are provided on the ABINIT web site.

The large number of platforms on which ABINIT has been installed has been made possible thanks to the use of cpp directives, coupled with MAKE files and/or different scripts. The unique set of ABINIT source files is preprocessed on-the-fly at compilation time, and generates machine-dependent code. As a result, ABINIT can work under UNIX-type OS, as well as under Windows and MacOS.

For each platform on which ABINIT has been installed, one machine-dependent file had to be prepared, containing the compiler name, options, and selected information needed for preprocessing (about 15 lines). These machine-dependent files depend weakly on the ABINIT version: once such a file has been set up for a platform, further installation on the same platform is quite simple.

The same software techniques are used to maintain the sequential and parallel versions of ABINIT, both produced from the preprocessing of a unique set of source files.

The MPI version of ABINIT has been tested (and used for production work) on clusters of SMP nodes of all the above-mentioned platforms working under UNIX-like OS, except Sun UltraSparc II. The OpenMP version of ABINIT has been tested on the following SMP machines: SGI Origin 2000, IBM RS6000 44P with Power 3+ processors (4 processors per node), and Compaq/DEC ES40 EV67 (4 processors per node).

3.4. Portability of automatic tests

The interplay between portability and automatic testing raises interesting problems. As mentioned in Section 3.2, the analysis of the result files involves comparison with reference files. These reference files are produced by runs on one specific platform. Although the automatic analysis tool "fldiff" is able to place a tolerance level on the floating point results (in this analysis, two floating point numbers are considered identical if they differ by less than 1.0e−12), which is sufficient to eliminate most of the platform variability, additional care in designing the tests of the ABINIT output was needed. Without it, a large fraction of the test cases would generate output files containing extensive platform-dependent sections, inappropriate for automatic analysis.

A first cause of largely different numerical results comes from the diagonalisation of matrices. Supposing some eigenvalues are degenerate, the corresponding eigenvectors can be chosen arbitrarily within the degenerate vector subspace. Even in the non-degenerate case, the phase of the wavevectors is undefined. Most of the computed properties (e.g. charge density, total energy) do not depend upon the particular choice of vectors or phases. However, this is not always true. Indeed, we are interested in:

(1) the output and visualisation of the atomic eigendisplacements generated by diagonalising the dynamical matrix;
(2) the initialisation of electronic wavefunctions (in order to help the convergence) from wavefunctions generated in (slightly) different geometries, often more symmetric.

In these two cases, specific sections of the output file might be completely different due to the choice of a phase or equivalent linear combinations of vectors, precluding any reasonably simple automatic analysis.

The ABINITv3.0 remedy to this first problem involves first (routine fxphas.f), fixing the phase of the eigenvectors, by maximising the (phase-dependent) sum of the square of their real parts, then choosing the sign of the first non-zero element to be positive, and second (routines phfrq.f and mkkin.f), slightly breaking the symmetry of the dynamical and kinetic energy matrices, at the level of one part per 1.0el2.

A second cause of portability problems comes from algorithms used in intrinsically unstable regimes. As an example, the optimisation of the interatomic distance based on the Broyden algorithm [61] involves the (implicit) inversion of the Hessian matrix. If one starts the optimisation of the distance between two atoms in a molecule in the region where the second derivative of the interatomic potential nearly vanishes, very small platform-dependent differences will be amplified, so that floating point results do not stay within the required tolerance.

The remedy to this second problem, set up in ABINITv3.0, is to design our test cases so as to avoid these regions, and if this proves impossible, to make the algorithm stop after only one or two iterations, when the results have not yet diverged on different machines.

A third cause of portability problems comes from sorting real numbers. Suppose that in a list of mostly unequal numbers, two numbers are equal. If this situation is not appropriately taken care of, the sorting algorithm might sort them in a systematic way, but that could vary on different platforms.

The physical realisations of this problem encountered in ABINITv3.0 are:

(1) producing a list of neighbouring atoms ordered by their distance with respect to some atom (symmetries often give atoms at the same distance), routines bonds.f and rsiaf9.f;
(2) producing a list of k-points ordered with respect to their magnitude (in order to fold to the irreducible Brillouin zone efficiently), routine listkk.f;
(3) finding the point of highest or lowest density, for printing purposes, routine prtrhomxmn.f.

The remedy to this third problem is to allow the algorithm to recognise that two numbers are equal, within a given tolerance, and to retain the ordering of the original list (before sorting) for these two numbers.

A fourth cause of portability problems is not directly related to physics. Suppose that some floating point number must be printed. If it is truncated at a precision less than the tolerance of our automatic analysis tool (e.g. using the f8.2 FORTRAN write format), depending on the closeness of the number to certain pivot, the rounding might vary on different machines.

The remedy to this fourth problem, would be to apply machine-independent rounding functions before printing the numbers. However, we have not yet applied it in ABINITv3.0.

Even after having solved these four problems, there are still a few test cases in which the floating point numbers differ by more than the selected tolerance in the result and reference files. These test cases usually involve response function calculations, in which the small lack of portability of the preliminary ground-state calculation are amplified by the subsequent response-function calculation. The observation of this lack of portability does not mean that some bug has been introduced. However some (human) knowledge of the meaning of the different sections of the output file is needed to assess the presence or the absence of abnormal behaviour.

Despite this remaining lack of portability, the examination of the summary produced by the automatic analysis tool for a complete set of tests takes at worst 2 or 3 min.

3.5. Self-documentation

The software concept of self-documentation is often mentioned when one speaks about maintaining large scientific codes, such as ABINIT. Let us first note that the documentation about one routine should be local to the routine, or accessible through a link local to the routine, in order to allow the maintenance of these routines by developers without global knowledge of the whole package. This idea is implemented in the "literate programming" approach [89]. Self-documentation tools allow developers to interlace program documentation with source codes, without having to maintain two separate documents. Users can grasp the whole structure of a the code in an automated manner: global documentation is generated from local documentation. Some tags must be used, at the local level, to provide guidance for the automatic tool.

The automatic tool could generate HTML files, with hyperlinks, such that the header of each routine could be printed on screen, with the description of what it does, the name of routines that call it ("parents") and the name of routines that are called by it ("children"). The index of all routines could be created. A search tool could also be created. Many other features such as an automated formatting of mathematical equations could be envisioned as well. To implement the above features systematically, developers must use and adhere to a standard format for their comments. Since this is typically seen as an extra burden, the requested effort should be kept as small as possible.

In ABINITv3.0, different auxiliary programs are used to implement this concept.

The first of these programs, "ROBODoc" [90], is a documentation tool that extracts specially formatted comment headers from the source file. It produces HTML (documentation) files automatically from the source text for each subroutine, including hyperlinks to access other subroutines' HTML files, even across directories. ROBODoc works with many languages, including FORTRAN90. The ABINIT developer must include a standardised header for every function, containing all sorts of information about that procedure/ function. Then, ROBODoc can create index tables for all variables, classes, functions, etc.

However, ROBODoc does not provide automatic listings of parent subroutines, although if such a list were provided, the hyperlinks to them would automatically be set up by ROBODoc. A PERL script called "parents" has been devised as to insert the list of parent routines in each routine automatically, just before ROBODoc processing.

Mathematical equations are another problematic issue with ROBODoc. It is partly resolved by employing the freeware "src2tex" [91], which can produce LaTeX files from source files for each subroutine, with nicely formatted equations. The formatting of equations into LaTeX style in each routine is part of the ABINIT style. The format for src2tex can be used simultaneously with ROBODoc [92].

In ABINITv3.0, the source code is formatted for ROBODoc and src2tex so that HTML/LaTeX files can be prepared for on-line/off-line documentation. In future versions, it is planned to make other documents written in LaTeX available as HTML files thanks to converters such as TtH [93]. Then the converted HTML files will be linked with the HTML documentation files of the source code.

3.6. User self-learning

Another interesting aspect of ABINITv3.0 concerns the possibility to learn how to use it without having to contact the developers, or to follow a school in a specified (geographic) location. For that purpose, tutorials have been set up. Five lessons, of 2 h each (including the time for the computations on a PC Pentium III at 800 MHz), written in HTML, guide the student in his/her first steps.

In the first two lessons, the student is familiarised with the computation of the formation energy, electronic structure and optimised interatomic distance of the hydrogen molecule, using different algorithms and exchange-correlation functionals. The third lesson introduces solid-state concepts, through the computation of the electronic structure of silicon, the paradigm of an insulator. The fourth lesson focuses on metallic aluminum, and ends with the computation of the aluminum surface energy. The fifth lesson deals with the dynamical and dielectric properties of AlAs (an insulator): phonons at Gamma, dielectric constant, Born effective charges, LO–TO splitting, phonons in the whole Brillouin zone.

This basic introduction to the use of ABINIT is completed by the availability of on-line manuals and help files, describing all the input variables in detail, as well as the availability of the input files used in the automatic testing, which provide examples of the use of all the ABINIT functionalities.

We have tried to simplify the use of ABINIT as much as possible. The organisation of the input data relies upon two files. In the first one, called "files" file, the user specifies the name of input and output files, including the pseudopotentials. It is rather short, usually six or seven lines long. In the other input file, the user specifies the values of all input variables for which the default value is to be overridden. This input file is parsed by ABINIT, which checks the existence of input variable keywords. There are more than 150 input variables, but usually less than 20 of them need to be mentioned in this file. A metalanguage allows specification of different sets of input variables in a single input file, and even to chain them in order to initialise some computation on the basis of the result of a previous computation in the same run.

3.7. Protocol for collaborative work

The concepts of free software, self-testing, self-documentation and user self-learning are obviously important for international collaborative development. It is also worth to organise the synchronisation of the work.

At the level of a single group of developers, we encourage the use of the CVS software [94] for developing ABINIT. However, all the developers must have access to the CVS repository for this software to be able to synchronise development efforts. Moreover, consultations between developers are still heavily needed.

The SourceForge [95] web site allows the transposition of the CVS ideas to the global scale: people from different countries, in different groups, can have access to the same repository. We have tried to use the SourceForge repository, in November 2000, but the access to it was found to be too slow for a project of the size of ABINIT. Thus, the protocol for collaborative development that we have followed since 1997 is still in effect.

We work in an iterative scatter–gather model: in the first step of the iteration, an official ABINIT version is released on the ABINIT web site; after a short period of time, the list of non-overlapping on-going development projects from the different groups is set up and published; each group works for two or three months on the part of ABINIT that has been temporarily attributed to it; then the contributions (source modifications and test cases) are gathered; two or three weeks are needed for the contributions to be merged (resolving possible conflicts using CVS), then tested and ported on at least five different platforms, before the release of a new official ABINIT version.

Finally, we would like to mention the existence of a coding style for ABINIT. In an effort to homogenise the coding style, we have explicitly written out the rules to be followed by the different groups of developers in a document called "coding_rules". This file is available in the ABINIT package, and is often updated. It is built upon the experience acquired during several years of development (including the experience of portability across platforms), and tries to build in ways to avoid generating certain classes of bugs. As an example, the use of "implicit none" is required in all FORTRAN routines.

The 10 different sections of the "coding_rules" concern the style to be followed for the declaration of variables, the choice of variable names, the FORTRAN source file format (in particular this format must allow processing by ROBODoc), the constructions for flow control, the use of arrays, some general good coding practice, specific coding rules for exception handling, old-fashioned practice—to be avoided—the use of BLAS and LAPACK subroutines, and topics of current reflection.

4. Discussion

Compared to the stated goals for the ABINIT project, much work still has to be done on the basis of ABINITv3.0. We describe now the functionalities that are currently under implementation, then the functionalities that are still lacking, and finally different software engineering concepts that have been only partially used, or are missing. Different groups, beyond the current list of authors, are contributing to these projects. A quite accurate view of the current status of the project is naturally available on the site [20].

The most ambitious implementation projects relate to:

(1) the use of the projector augmented wave technique of Blöchl [13] and ultrasoft pseudopotentials [10];
(2) interfacing ABINIT with a GW code;
(3) the treatment of spinor wavefunctions (including SO) for the non-collinear magnetism case;
(4) the generalisation of response-function computations to all cases presently available for ground-state computations;
(5) band-by-band parallelisation, and the optimisation of the current imple-mentated parallelism, especially when different levels of parallelisation are used concurrently;

Other development efforts concern:

(1) the recognition of spatial space group numbers;
(2) the $2n + 1$ theorem of density functional perturbation theory [71];
(3) Kohn–Sham exact exchange [86–88];
(4) Shubnikov (anti-ferromagnetic) symmetry groups;
(5) the treatment of surface dipoles;
(6) frequency-dependent conductivity;
(7) further features related to the Berry phase computation of the polarisation [96];
(8) the automatic generation of lattice Wannier functions.

On the other hand, work is needed in the following aspects, but not yet scheduled:

(1) real-space treatment of the non-local operators [97];
(2) decomposition of the density-of-states in atomic orbital components;

(3) response to elastic and alchemical perturbations;
(4) new methods to optimise geometries, and the computation of transition states.

At the level of software engineering, first there is an ongoing effort to deal correctly with equations, in the self-documentation. Then, derived datatypes (scarcely present in ABINITv3.0) should be used. The object-oriented features of FORTRAN90 are also not used in ABINITv3.0: the object-oriented FORTRAN90 capabilities are not as developed as in C++, but they should be incorporated. The F90 limitation in object-oriented features brings us of course to the possible limits of the present ABINIT basic options: for historical reasons (reuse of existing code), we have built a FORTRAN90 code. As an alternative to C++, many current open source projects use Python scripting language coupled with another language for compute-intensive tasks (usually C, but why not F90). We still think that the basic options of our project should make it viable and fruitful for many years.

We feel that the ABINIT project is already successful, in that a large number of capabilities have been built in the relatively small time since the beginning of the project. The organisation of the collaborative work is such that many people were able to contribute. We actually followed the example given in the computer science community for the development of Linux [98].

While this article does not focus on using ABINIT to actually get useful calculations done, the purpose of the project is of course to have a reliable state-of-the-art computer program for predicting properties of materials using ab initio methods. In that light, there are over 70 publications listed on the ABINIT web site [20] showing the wide scope of applications and the many individuals and groups that have used the code so far. The growth of useful output is our primary measure of success.

Acknowledgements

We would like to thank first the advisory committee of ABINIT, including, beyond some of the authors of this paper, N. Hill, K. Rabe, L. Reining, and D. Teter. We also thank other contributors to the ABINIT software project, ABINITv3.0 or later versions, or people who have given advice and warm support: A. Alouges, S. Bernard, J.-C. Charlier, T. Deutsch, S. Erwin, A. Filipetti, R. Godby, S. Goedecker, A. Horsfield, A. Khein, D.D. Klug, C. Lee, Z. Levine, B. Magne, H. Matthis, J.-P. Minet, V. Olevano, G. Onida, A. Quandt, Na Sai, M. Stengel, J. Sullivan, P. Tepesch, P. Thibaudeau, J. Tse, D. Vanderbilt, B. Van Renterghem, M. Veithen, A. Zaitsev. We (X.G., J.-Y.R., and G.-M.R.) thank the national fund for scientific research (Belgium) for financial support, as well as computational resources (FRFC project NB 2.4556.99). We also acknowledge financial support from the "Pole d'attraction interuniversitaire" (Phase 5—Federal government of Belgium), and the "Action de recherche concerte—interaction electron-vibration dans les nanostructures".

References

[1] P. Hohenberg, W. Kohn, Phys. Rev. 136 (1964) B864.
[2] W. Kohn, L.J. Sham, Phys. Rev. 140 (1965) A1133.
[3] E. Runge, E.K.U. Gross, Phys. Rev. Lett. 52 (1984) 997.
[4] R. Car, M. Parrinello, Phys. Rev. Lett. 55 (1985) 2471.
[5] L. Hedin, Phys. Rev. 139 (1965) A796.
[6] M.S. Hybertsen, S.G. Louie, Phys. Rev. B 35 (1987) 5585.
[7] S. Baroni, P. Giannozzi, A. Testa, Phys. Rev. Lett. 58 (1987) 1861.
[8] D.C. Allan, M.P. Teter, Phys. Rev. Lett. 59 (1987) 1136.
[9] L. Kleinman, D.M. Bylander, Phys. Rev. Lett. 48 (1982) 1425.
[10] D. Vanderbilt, Phys. Rev. B 41 (1990) 7892.
[11] R.D. King-Smith, D. Vanderbilt, Phys. Rev. B 47 (1993) 1651.
[12] K.D. Brommer, M. Needels, B.E. Larson, J.D. Joannopoulos, Phys. Rev. Lett. 68 (1992) 1355.
[13] P. Blöchl, Phys. Rev. B 50 (1994) 17953.
[14] M.E. Casida, in: J.M. Seminario (Ed.), Recent Developments and Applications of Modern Density Functional Theory, Elsevier, Amsterdam, 1996.
[15] M. Petersilka, U.J. Gossman, E.K.U. Gross, Phys. Rev. Lett. 76 (1996) 1212.
[16] L.X. Benedict, E.L. Shirley, R. Bohn, Phys. Rev. Lett. 80 (1998) 4514.
[17] S. Albrecht et al., Phys. Rev. Lett. 80 (1998) 4510.
[18] M. Rohlfing, S. Louie, Phys. Rev. Lett. 81 (1998) 2312.
[19] http://www.gnu.org.
[20] http://www.abinit.org.

[21] http://www.cscs.ch/projects/CP2K.html or http://cp2k.berlios.de.
[22] http://www.fysik.dtu.dk/CAMP/CAMPOS_welcome.html.
[23] http://www.pwscf.org.
[24] http://cst-www.nrl.navy.mil/people/singh/planewave.
[25] http://www.cineca.it/∼acvO/CP/carpar.html.
[26] http://www.wfu.edu/∼natalie/papers/pwpaw/man.html.
[27] http://www.fhi-berlin.mpg.de/th/fhimd.
[28] http://cms.mpi.univie.ac.at/vasp.
[29] http://www.tcm.phy.cam.ac.uk/castep.
[30] http://www.pt.tu-clausthal.de/∼ptpb/PAW/pawmain.html.
[31] http://www.nersc.gov/projects/paratec.
[32] http://www.gnu.org/copyleft/gpl.txt.
[33] M. Payne, B.C. Allan, T. Arias, J. Joannopoulos, Rev. Mod. Phys. 64 (1992) 1045.
[34] J. Perdew, A. Zunger, Phys. Rev. B 23 (1981) 5048.
[35] S. Goedecker, M. Teter, J. Mutter, Phys. Rev. B 54 (1996) 1703.
[36] J. Perdew, K. Burke, M. Ernzerhof, Phys. Rev. Lett. 77 (1996) 3865.
[37] R. van Leeuwen, E. Baerends, Phys. Rev. A 49 (1994) 2421.
[38] O. Gunnarsson, B. Lundqvist, Phys. Rev. B 13 (1976) 4174.
[39] E. Wigner, Trans. Faraday Soc. 34 (1938) 678.
[40] J. Slater, Phys. Rev. 81 (1951) 385.
[41] N. Troullier, J. Martins, Phys. Rev. B 43 (1991) 1993.
[42] C. Hartwigsen, S. Goedecker, J. Hutter, Phys. Rev. B 58 (1998) 3641.
[43] M. Teter, Phys. Rev. B 48 (1993) 5031.
[44] M. Fuchs, M. Scheffler, Comput. Phys. Commun. 119 (1999) 67.
[45] http://bohr.inesc.pt/jlm/pseudo.html.
[46] S. Louie, S. Froyen, L. Cohen, Phys. Rev. B 26 (1982) 1738.
[47] E. Pehlke, M. Scheffler, Phys. Rev. Lett. 71 (1993) 2338.
[48] H. Monkhorst, J. Pack, Phys. Rev. B 13 (1976) 5188.
[49] M. Teter, M. Payne, D. Allan, Phys. Rev. B 40 (1989) 12255.
[50] P. Dederichs, R. Zeller, Phys. Rev. B 28 (1983) 5462.
[51] D. Anderson, J. Assoc. Comput. Mach. 12 (1964) 547.
[52] X. Gonze, Phys. Rev. B 54 (1996) 4383.
[53] J. Annett, Comput. Mater. Sci. 4 (1995) 23.
[54] H. Kai-Ming, J. Ihm, J. Joannopoulos, Phys. Rev. B 25 (1982) 4260.
[55] C.-L. Fu, K.-M. Ho, Phys. Rev. B 28 (1983) 5480.
[56] M. Methfessel, A. Paxton, Phys. Rev. B 40 (1989) 3616.
[57] N. Marzari, Ph.D. thesis, University of Cambridge, 1996, http://alfaromeo.princeton.edu/marzari/preprints/.
[58] T. Hahn (Ed.), International Tables for Crystallography, Reidel Publishing Company, Boston, 1983.
[59] H. Hellmann, Einfuhrung in die Quantumchemie, Deuticke, Leipzig, 1937.
[60] R. Feynman, Phys. Rev. 56 (1939) 340.
[61] C. Broyden, Math. Comput. 19 (1965) 577.
[62] B. Schlegel, J. Comput. Chem. 3 (1982) 214.
[63] C. Wang, Q.-M. Zhang, J. Bernholc, Phys. Rev. Lett. 69 (1992) 3789.
[64] L. Verlet, Phys. Rev. 159 (1967) 98.
[65] F.S. Acton, in: Numerical Methods That (Usually) Work, Harper and Row, New York, 1970, p. 130.
[66] S. Nosé, Mol. Phys. 52 (1984) 255.
[67] S. Nosé, J. Chem. Phys. 81 (1984) 511.
[68] S. Nosé, Mol. Phys. 57 (1986) 187.
[69] W. Hoover, Phys. Rev. B 31 (1985) 1695.
[70] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Oxford University Press, New York, NY, 1990.
[71] X. Gonze, J.-P. Vigneron, Phys. Rev. B 39 (1989) 13120.
[72] X. Gonze, Phys. Rev. A 52 (1995) 1086.
[73] X. Gonze, Phys. Rev. B 55 (1997) 10337.
[74] X. Gonze, C. Lee, Phys. Rev. B 55 (1997) 10355.
[75] http://www.mathworks.com.
[76] http://www.cscs.ch/molekel.
[77] L. Adler, Phys. Rev. 126 (1962) 413.
[78] N. Wiser, Phys. Rev. 129 (1963) 62.
[79] F. Jollet, M. Torrent, G. Zerah, X. Gonze, unpublished.
[80] L. Kleinman, Phys. Rev. B 21 (1980) 2630.
[81] L.A. Hemstreet, C.Y. Fong, J.S. Nelson, Phys. Rev. B 47 (1993) 4238.
[82] D.C. Langreth, J.P. Perdew, Solid State Commun. 17 (1975) 1425;
Phys. Rev. B 15 (1977) 2884.
[83] E.K.U. Gross, J.F. Dobson, M. Petersilka, Density functional theory II, in: R.F. Nalewajski (Ed.), Topics in Current Chemistry, vol. 181, Springer, Berlin, 1996, p. 81.
[84] J.F. Dobson, J. Wang, Phys. Rev. Lett. 82 (1999) 2123.
[85] M. Fuchs, X. Gonze, unpublished.
[86] R.T. Sharp, G.K. Horton, Phys. Rev. 90 (1953) 317.
[87] J.D. Talman, W.F. Shadwick, Phys. Rev. A 14 (1976) 36.
[88] A. Görling, M. Levy, Phys. Rev. A 50 (1994) 196.
[89] http://www.literateprogramming.com/ and the links therein.
[90] http://www.xs4all.nl/∼rfsber/Robo/robodoc.html.
[91] http://www.eng.gunma-u.ac.jp/∼amano, version 2.13j.
[92] Though the src2tex software has to be slightly modified.
[93] http://hutchinson.belmont.ma.us/tth/ TtH is useful to make HTML documents with mathematical equations without relying on any picture files.
[94] http://cvshome.org/.
[95] http://www.sourceforge.net.
[96] R. Nunes, X. Gonze, Phys. Rev. B 63 (2001) 115118.
[97] R.D. King-Smith, M.C. Payne, J.S. Lin, Phys. Rev. B 44 (1991) 13063.
[98] http://www.tuxedo.org/∼esr/writings/cathedral-bazaar/.