NFP Manual 1.01

October 10, 1997

The author can be contacted as follows:

M. Methfessel
Institute for Semiconductor Physics
Walter-Korsing Str. 2, D-15230 Frankfurt (Oder), Germany
`methfessel@ihp-ffo.de`

The nfp program package described here was developed by M. Methfessel and M. van Schilfgaarde. The manual was written by M. Methfessel. MvS was supported by the Office of Naval Research.

Contents

Chapter 1

Preliminaries

Purpose of This Document

This document describes the "NFP" full-potential LMTO program for calculating the electronic structure of crystalline systems. The principal aim is that, with the help of this document, a user can set up and run calculations to a large extent on his own.

Who Should Read This Document?

Three groups of users are adressed. The first contains those who already have used the LMTO method or another electronic-structure programs in the past. The second group contains those who have no such previous experience but would like to perform a calculation using this program. The third group consists of those who are interested in adapting parts of the program for their own purposes. For these reasons, in addition to the instructions on how to run the program, elementary topics are discussed briefly and a reasonably complete description of the program internals is given.

Background

Calculations based on density-functional theory are more in the nature of "simulation" than "theory." Their aim is a realistic and detailed simulation of the bonding in condensed-matter systems. Due to the central significance of orbital structure for chemical bonding, some treatment of the electronical quantum mechanics is unavoidable. Density-functional theory supplies a practical recipe by which this can be done. The result is a workable procedure which can supply the total energy of a reasonably large system, starting only from the positions and the types of the nucleii. By exploring the total-energy surface as the nucleii are placed in different positions, a large amount of useful data can be obtained. For example, the equilibrium structure can be determined by minimising the energy, or a diffusion barrier can be obtained by plotting out the energy along a diffusion path.

As in other simulation methods, a set of coupled equations must be solved to high numerical precision. In the present case, the Schrödinger equation for non-interacting electrons in a realistic potential and the Poisson equation to obtain the the electrostatic potential are major challenges. Different methods have been developed to handle these tasks. It is fair to say that none of the methods can be treated completely as a black box. In each case, some knowledge of the underlying ideas is needed in order choose input parameters reasonably and in order to interpret the results. Equally important, a user must recognize when things start to go wrong and should be able to do something about it. The intention is to present the required information as clearly and succinctly as possible here.

Among the different methods, LMTO has the reputation of being mildly complicated anyway, and this program adds some completely new features to the existing lore. The first important modification is to define the basis functions using non-singular "smooth" Hankel functions. These are more similar to the true wavefunctions and can be integrated efficiently using a real-space mesh. The second feature is a new formulation of the all-electron approach, which has some similarity to the pseudopotential procedure. The gains are simpler expressions, lower angular-momentum cutoffs, and a straightforward force theorem. A user of this program should have some idea of these two features.

Current State of the Project

This is the first version of the program which is distributed. Possibly it will soon be replaced by a newer version when feedback comes in. The features currently implemented are:

- Self-consistent calculations for general geometries, without need for empty spheres, and for any type of atom.
- Forces on the atoms, *i.e.*, the partial derivative of the energy surface.
- Scalar-relativistic variant.
- Frozen overlapped core approximation (FOCA). This is intended to supersede the former "two-panel" calculations. The core electron den-

sity in the crystal is made by overlapping the frozen free-atom cores.

These features are still missing but will come Real Soon:

- Spin-polarized variant.
- Gradient corrections.
- Something which uses the forces, *i.e.*, simulated annealing or steepest descent to optimize the geometry.

A problem which has not yet been solved in this version is to construct a good starting density at a shifted geometry, given the selfconsistent density at some set of positions. Currently, it is often more efficient to restart from the overlapped free-atom density whenever going to a new geometry. It is more or less a prerequisite to find a solution for this problem before the program is adapted to perform simulated annealing or molecular dynamics.

Chapter 2

Basics

This chapter summarizes some features of density-functional theory, purely from a user's viewpoint and without any attempt at a decent presentation of the formalism. Secondly, it characterizes the standard LMTO approach in comparison to other methods to solve the equations of density-functional theory.

2.1 Minimal Density-Functional Theory

The subject of DFT is the interaction between the electrons in a condensed-matter system. Due to the strong Coulomb repulsion between electrons, this makes a contribution to the total energy which can in no way be ignored. Unfortunately, the formally correct approach (calculation of the many-body wavefunction for the coupled electrons) is generally not feasible. For the practician, the beauty of DFT and the local-density approximation (LDA) is that they handle this problem by a method which reduces to a simple recipe. In the end, we are given a practical method to calculate total energy as function of the nuclear positions, which is the aim of the whole procedure. A good review of density-functional theory is Ref. [1].

2.1.1 The DFT Recipe

By invoking DFT, in the end we must solve the *single-particle* Schrödinger equation for enough eigenstates to take up all the electrons. This task is still formidable but it is at least within the realms of human endeavour, even for systems containing up to a few hundred atoms. The electron-electron interaction is included because the potential entering the Schrödinger equation depends on the total electron density. This “effective potential” has three parts: the external potential from the nucleii, an electrostatic potential calculated by solving the Poisson equation for the total electron density, and an "exchange correlation" potential which essentially weakens this second term. Most calculations use the extremely successful "local-density approximation", and then the exchange-correlation potential is just a simple tabulated function of the local electron density. It is, in fact, rather close to a negative constant times the third root of the density.

In this formalism, the electron density depends on the effective potential (via the single-particle Schrödinger equation) which in turn depends on the density (via the Poisson equation and the exchange-correlation term). Thus, a coupled system of equations must be solved simultaneously. Once this has been done, the total energy is assembled in a similar manner as the potential. It is the sum of the electron kinetic energy, the electrostastic energy, and an exchange-correlation contribution.

A calculation using the LDA is "ab-initio" because no other information is put in except the types of the atoms and where these are positioned. It follows that calculations done by different groups for the same system must yield essentially the same results. Any differences must be the outcome of technical problems. Solving the LDA equations accurately is a major challenge, so such differences are not unknown.

Why it works

Here is a hand-waving explanation for the success of the simple DFT/LDA recipe, which can be formulated more rigorously within DFT. Roughly following Ref. [2], we discuss what potential a typical electron in the crystal feels. It is natural to start with the attractive potential of the nucleii, then to add on the the repulsive potential of the electron distribution. Together this gives the electrostatic potential of the complete system, in the way that an infinitesimal external test charge would sample it. This procedure is called the Hartree approximation, and it does not lead to useful results in the context of electronic-structure calculations.

What is missing here is the following insight: if the electron in question is known to be at some point $\mathbf{r}$, the other $N-1$ electrons tend to stay away from this point. This happens for two reasons, namely the direct Coulomb repulsion (correlation) and the antisymmetry of the many-body electron wavefunction (exchange). In the present context, the relevant electron distribution is that of the $N-1$ other electrons under the condition that the first electron is known to be at $\mathbf{r}$. This looks like the full electron density in most parts of space, but near point $\mathbf{r}$ it is reduced, making an "exchange-correlation" hole of one electron. Consequently we can obtain the correct potential by subtracting the electrostatic potential due to the charge which was removed to make the hole. The practical problem is, of course, that the shape of this hole is not known; calculating it is equivalent to solving the (unsolvable) many-body problem. Therefore, we wildly estimate the size and shape of the hole by inspecting the homogeneous free-electron gas with the same density as our system has at point $\mathbf{r}$ (this is the local-density approximation, LDA). Overall, the most important effect to be included is that the exchange-correlation hole becomes compressed for higher densities and vice versa. This effect is described adequately within the LDA, leading to the success of the method.

2.1.2 Summary of Important Properties

Density-functional theory was designed to calculate the total energy. As a side product, it produces the electron density. Keeping this in mind, there are a few properties of the DFT/LDA of which a user should be aware of:

- Binding energies are calculated as the difference of the total energies for the bound system and its separated constituents. These usually come out somewhat too large. Typical values for the overbinding are 1.5 eV of about 9.8 eV for the N_2 dimer, and 0.7 eV of 3.4 eV for the cohesive energy of fcc aluminum. Gradient corrections, which are a bit of a patch on top of the LDA, can improve this substantially.

- The geometry is calculated by minimising the total energy respective to the atom positions. Here agreement to experiment is very good; bond lengths are generally slightly too short but the error is only a few percent.

- Vibrational frequencies calculated from the curvature of the energy surface near the minimum, also agree quite well.

- To use the local-density approximation, information about the free-electron gas is needed, as tabulated in the exchange-correlation parametrisation. The free-electron gas is in itself a complicated problem and slightly different parametrisations of the LDA potential exist, such as the Hedin-Lundqvist and Ceperly-Alder variants. Total energies can be somewhat different for these, especially if core states are included, but energy differences are insensitive. The choice of the XC potential can have some effect on subtler quantities such as the surface energy.

- Since a single-particle Schrödinger equation being is solved, eigenvalues come out. For a crystal, these can be plotted as a function of the wave vector to make a bandstructure. DFT emphatically *does not* claim this is the band structure as it would come out from spectroscopic measurements. Formally, the calculated eigenvalue in DFT gives the change in the total energy (per added charge) when an infinitesimally small amount of charge is added to the corresponding eigenstate, something which can easily be done in DFT but not in nature. The point is that experiments always add or remove one full electron. Depending on the system, the calculated eigenvalues can have some bearing on the experimental results or not. In general terms, the usefulness of the LDA bandstructure when interpreting experiments is smaller for more strongly correlated systems.

- As a specific instance of the previous point, calculated band gaps for semiconductors and insulators are invariably too small. For silicon, the measured and calculated gaps are 1.17 and about 0.6 eV. At the risk of belaboring the obvious, this "error" is not a problem of the LDA method but is due to a wrong interpretation of its output. Extensions to the LDA, notably the GW approximation [3], have been developed to calculate band gaps and other excitation energies.

- For some cases, excitation energies can be obtained within the LDA. This happens when the system before and after removal of an electron can both be "handled within the formalism." For example, core-level binding energies can be calculated to surprising accuracy by removing an electron and recalculating the total energy. For atoms, ionicities and electron affinities can be calculated, although these show some systematic deviations from measured values.

- For heavy atoms, relativistic effects start to play a role. If spin-orbit coupling is neglected, these can be included within the scalar-relativistic equation [4], which adds some terms to the Schrödinger equation. One important effect is that s states are pulled down relative to p and d states. For lighter atoms (say, up to the second-row transition metals), relativistic effects can change the total energy of a system possibly significantly, but binding energies and other energy differences will change only slightly. Generally, bond lengths will shorten by a small amount. For all atoms further down in the periodic table, scalar-relativistic treatment is unavoidable.

- The LDA has been extended to magnetic systems in a spin-polarized version. In the local spin-density approximation (LSDA) there are two separate density distributions for the spin-up and spin-down electrons. These feel two separate effective potentials, which are different only in the exchange-correlation terms. In practical calculations, the spin-up and spin-down electrons are treated quite independently of each other except in the step where the potential is made and the total energy is evaluated. Since most free atoms have a non-zero total spin, they must be treated with a spin-polarized program. This must be remembered when calculating the binding energy of a system relative to the free atoms.

2.1.3 LDA in practice

What steps must be done by a practical implementation of the LDA? The electron density depends on the potential and the potential on the density, so a coupled set of equations must be solved. These involve the wave functions $\psi_i(\mathbf{r})$, the electron density $n(\mathbf{r})$, and the effective single-particle potential $V_{\text{eff}}(\mathbf{r})$:

$$V_{\text{eff}}(\mathbf{r}) = V_{\text{N}}(\mathbf{r}) + 2\int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\, d\mathbf{r}' + \mu_{\text{xc}}(n(\mathbf{r})) \tag{2.1}$$

$$\left[-\tfrac{1}{2}\nabla^2 + V_{\text{eff}}(\mathbf{r})\right] \psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}) \tag{2.2}$$

$$n(\mathbf{r}) = \sum_i \psi_i^*(\mathbf{r})\psi_i(\mathbf{r}) \ . \tag{2.3}$$

The first equation assembles the effective potential out of the nuclear, electrostatic (Hartree) and exchange-correlation contributions, given the input electron density. The second equation is the single-particle Schrödinger equation for the effective potential, to be solved for enough eigenstates to take up all the electrons. The third equation adds together the squared wavefunctions to make a new electron density.

To solve the equations simultaneously, we start with some guessed density and iterate, making alternatively a new potential and a new density. A very good starting guess is to overlap the electron densities of the free atoms. If the iterative procedure converges, we have found a fixed point of the prescription “go around the loop once, making the effective potential and a new output density” and therewith a solution to the set of equations. This is also known as “making the system selfconsistent.” Generally, the procedure will *not* converge unless some damping is used, *e.g.*, in the form

$$n_{i+1}^{\text{in}} = \beta n_i^{\text{out}} + (1-\beta) n_i^{\text{in}} \tag{2.4}$$

to make the new input density, using a mixing parameter β between zero and one.

Evaluating the Total Energy

While iterating, the total energy is calculated in each iteration. It is given as the sum of kinetic, electrostatic, and exchange-correlation contributions:

$$E_{\mathrm{KS}} = T_s + U + E_{\mathrm{xc}} \tag{2.5}$$

where U is the electrostatic energy of the complete systems including the nucleii and KS stands for "Kohn-Sham". The kinetic energy is the difference of the eigenvalue sum and the expectation value of the input potential:

$$T_s = \sum_i \epsilon_i - \int n^{\mathrm{out}} V_{\mathrm{eff}}\, d\mathbf{r}\ . \tag{2.6}$$

Terms U and E_{xc} are integrals involving the density and should be evaluated for the output density n^{out} in each iteration. Together this gives the Kohn-Sham energy for n^{out}. It should converge from above towards the final value as selfconsistency is approached.

In addition, we can evaluate the Harris-Foulkes energy [5] E_{H} in each step. This is done by replacing n^{out} by n^{in} everywhere in the two equations above while keeping the eigenvalue sum as it is. Like E_{KS}, the Harris energy converges quadratically, but it should approach the limit from below. In rare cases, this will not be the case, for example, when the basis set adapts to the changing potential. However, generally E_{H} and E_{KS} bracket the converged energy and inspecting both gives a good feeling for the convergence.

One should appreciate the high accuracy to which these equations must be solved. The energy differences of interest are often in the order of a few mRy, for example when moving the atoms to map out the energy change of a vibrational frequency. The total energies per atom are typically thousands of Rydbergs. It is a major effort to solve the Schrödinger and Poisson equations to this kind of accuracy in an efficient way. In practice, DFT calculations are much closer to a simulation method than to theories based on simple concepts such as tight binding.

Applying DFT to Crystalline Systems

The DFT/LDA formalism can be applied equally well to atoms, molecules, clusters, surfaces, interfaces, and so on. For atoms, the equations can be solved comparatively easily in spherical coordinates. For molecules and clusters, a full-blown apparatus is needed, capable of accurately treating potentials, densities, and wave functions of arbitrary shape.

For crystals, the problems faced by a molecular calculation are also present, plus some additional difficulties. Notably, the eigenstates appearing in Eqs. (2.2,2.3,2.6) must run over the (now complex) wavefunctions associated with the Bloch vector **k**, which scans through the first Brillouin zone. Formally, the discrete sums over the eigenstates should be replaced by integrals over the Brillouin zone combined with sums over the bands at each **k**. In practice, the integrals over the zone are replaced by sums over a suitable regular mesh of **k** points in practice. The fineness of this mesh is a parameter which should be chosen carefully; a coarse mesh leads to inaccurate results, while an overly fine mesh increases the computing time.

Special Problems for Metals

For metals, a severe additional problem appears because the integration variable **k** runs only over that part of the Brillouin zone (BZ) which lies inside the Fermi surface. The Fermi surface winds it way in some complicated fashion between the mesh points. It is definitely not good enough to simply sum over those mesh points which lie inside it. Two solutions are often used. Interpolation techniques, such as the linear tetrahedron method, interpolate between the mesh points to get an approximate representation of the energy bands for all points in the BZ, then integrate this representation accurately. Smearing methods essentially smear out the sharp cutoff at the Fermi surface, analagous to the state of affairs at higher temperatures. The integrands then revert to smooth functions defined over the whole BZ, amenable to a simple mesh summation. The problem here is to ensure that the errors due to the smearing are small.

In comparison, it is a smaller bookkeeping problem to perform the **k** space summation only over an irreducible symmetry wedge of the BZ, then to symmetrize the output density. This always saves a factor of two because the information contained in vectors **k** and $-\mathbf{k}$ is the same. For systems of high symmetry, the saving is much larger: in the cubic systems, the irreducible wedge is only 1/48th of the total Brillouin zone.

Using Supercells

A program which can handle reasonably large crystalline systems can be applied to calculate surfaces, intefaces, and impurities. These are among the most important applications of DFT currently. The key is the "supercell technique" which simulates a basically non-periodic geometry using a periodic unit cell. For a surface, a "slab" or thin film of a few atomic layers can be used, repeated infinitely with some empty space between adjacent slabs. If this empty space is large enough, adsorbate molecules can be placed on the surface. Similarily, an interface between, say, materials A and B can be describes by an infinite series of alternating thin A, B, A, B... layers. In this spirit, an impurity is modeled by a superlattice of regularily arranged impurities, spaced far enough to eliminate most residual interaction. Finally, atoms and molecules can be treated within a crystalline program by putting them into the center of a reasonably large, otherwise empty box.

Luckily, the supercells needed can often be quite small. A metal slab might be five to seven layers thick, and even a three-layer slab has some bearing on the true surface. Impurity supercells for something like a metal vacany might contain eight to sixteen atoms. For semiconductors, the supercells have to be chosen somewhat larger. Finally, molecules have negligible interaction when they are no closer than approximately three times the characteristic bond lengths.

2.2 The LMTO Basis Set

When simulating a condensed-matter system using density-functional theory, the main calculational steps are (i) to solve the Poisson equation and to add on the exchange-correlation potential, making the effective potential felt by an electron, (ii) to solve the single-particle Schrödinger equation, and (iii) to add together the squared wavefunctions to make the new output density. Of these, solution of Schrödinger's equation is the most difficult and the most expensive step computationally.

Different methods use different approaches to solve the Schrödinger equation. The most fundamental classification is into pseudopotential and all-electron methods. The former construct a smooth pseudopotential to take the place of the nuclear potential, eliminating the core wavefunctions from the problem. All-electron methods such as the linear muffin-tin orbital (LMTO) method [6] calculate the core eigenstates explicitly. This can happen either once in the beginning (in the frozen-core approximation) or repeatedly in each self-consistency iteration.

A second important classification is by the approach used to represent the valence wavefunctions. For smooth pseudopotentials, the obvious choice is a sum of plane waves. However, some atoms require deep pseudopotentials and then possibly prohibitively many plane waves will be needed. In contrast, all-electron methods must face the problem that the valence wavefunctions have strong localized oscillations near the nucleus, but have some general shape in the bonding region between the atoms. A good solution for this problem is called augmentation, first introduced by Slater some time ago [7]. We first choose a set of smooth "envelope" functions which can describe the wavefunction in the interatomic region. These are then modified in the vicinity of each nucleus to put in the proper oscillatory behavior. Different methods are distinguished by the envelope functions they use: the linear augmented-plane wave (LAPW) method [6] uses plane waves, whereas LMTO uses spherical Hankel functions. The latter are not as exotic as they sound; they are just spherical waves emitted from a point somewhere in space. A review of different methods can be found in Ref. [8].

2.2.1 Augmentation: LAPW and LMTO

In detail, augmentation works by cutting space into muffin-tin spheres centered on the various nucleii and an "interstitial region," which is the oddly-shaped region left between the spheres. Inside each atomic sphere, the analytical envelope function is replaced by a numerical solution of the Schrödinger equation which matches smoothly at the sphere surface. This solution can be calculated relatively easily because the potential is very close to spherical there, permitting a straightforward solution of the radial Schrödinger equation for the different angular momentum components. More precisely, in this context of defining the basis set, the potential near the nucleus is taken as spherical, but the non-spherical terms are included properly later on.

All-electron methods using augmentation are distinguished by the set of envelope functions they use. This choice is somewhat restricted by the task at hand. On the one hand, we must be able to calculate all the required quantities. Among these are the overlap integrals and the matrix elements of the Hamiltonian, and the modulus-squared of the wavefunction for the output density. On the other hand, the basis set should be as simple as possible (to permit implementation in a finite time) and small (hopefully leading to fast program execution and small storage requirements). The linear augmented plane-wave (LAPW) method uses plane waves as envelope functions. Each envelope function is spread homogeneously over the unit cell and is not associated with a specific site. A major advantage of this choice is simplicity. A disadvantage is that, depending on the system, a large number of basis functions will often be needed.

The linear-muffin-tin orbital (LMTO) approach is more complicated. The envelope functions are "solid Hankel functions" $H_L(\mathbf{r}) = h_l(\kappa r)Y_L(\hat{r})$, consisting of a radial Hankel function times a spherical harmonic of the angle. This object has a well-defined angular momentum $L=(l,m)$ and is centered at some specific atom in the crystal, where it has a singularity. The LAPW and LMTO basis functions are presented schematically in Fig. 2.1.

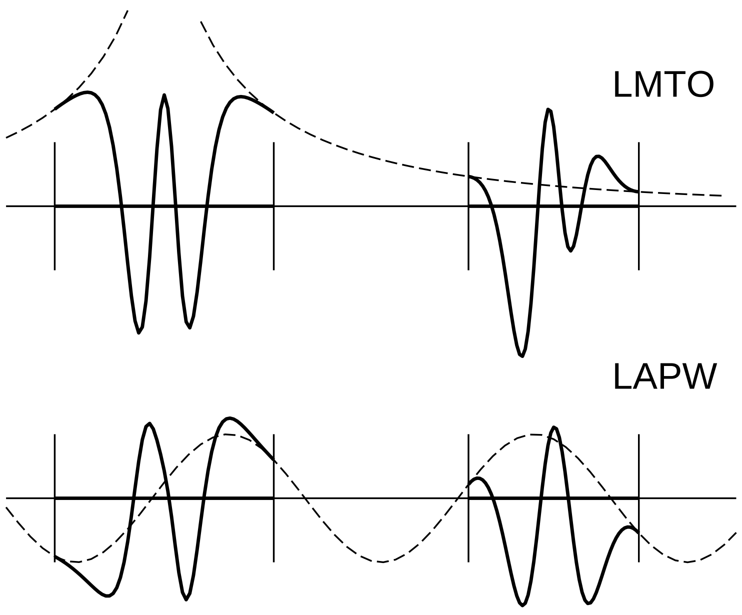

Figure 2.1: Qualitative sketch of the LMTO and LAPW basis functions. Both start from a smooth envelope function (shown dashed). The envelope is defined as an atom-centered Hankel function when making an LMTO and a plane wave in the case of an LAPW. Inside the atomic spheres (shown by thicker lines), the envelope functions are replaced by numerical solutions of the Schrödinger equation which match smoothly at the sphere boundaries.

2.2.2 Advantages and Disadvantages of LMTO

To the unsuspecting user, the advantages of defining the LMTO basis functions as augmented Hankel functions are not especially obvious. Such a more complicated definition leads to an involved formalism and larger programming effort. What are the arguments for doing it?

- The central argument is that the LMTO functions were constructed to be similar to the true crystal wavefunctions. In fact, if the crystalline potential is approximated by the muffin-tin form (i.e., spherical inside the spheres and flat outside them) the true crystal wavefunction turns out to be a finite sum of LMTO functions. Since the muffin-tin form is not such bad approximation, it can expected that the wavefunctions in the correct potential can be written as sums over only a few different LMTOs.

- As a consequence of the small basis size, calculations should be fast. More precisely, the dominant steps in a calculation scale as the third power of the basis size. All other things being equal, reducing the basis by half should save seven-eighth of the computer time. Of course, increased complexity in executing these steps can eliminate some or all of the gain.

- Another consequence of the smaller basis is reduced memory requirement, which can be equally important as savings in computer time when calculating large systems.

- The LMTO envelopes, *i.e.*, the solid Hankel functions, are rather simple analytical objects. This helps to perform the various steps which have to be done. Ultimately, a lot of the useful properties come about because these functions are eigenfunctions of the kinetic energy operator:
$$-\Delta H_L(\mathbf{r}) = \epsilon H_L(\mathbf{r}) \tag{2.7}$$
where $\epsilon = -\kappa^2$ is an energy which charcterizes the localization of the function.

- When chosing the basis set for a specific system, chemical intuition can be used. The basis can be tailored to the problem at hand, since it can be chosen for each atom separately, and the results can sometimes be interpreted in simpler terms due to the atom-oriented basis functions.

The following positive features are shared by the LAPW method:

- A major advantage is numerical stability in the context of solving the Schrödinger equation. Again, this comes about because each separate function is already a solution of the equation, albeit in a simplified potential. In practice, the basis set can be stocked up considerably beyond accurate convergence before stability problems (generally a non-positive-definite overlap matrix) set in. In contrast, the gaussian-based basis sets often used in quantum chemical calculations are more sensitive.

- The LMTO basis set can be applied equally well to all atoms in the periodic table. When including a new type of atom, no effort is needed to construct and test a suitable pseudopotential.

- As in other all-electron methods, data concerning the core states is available which cannot be directly supplied in a pseudopotential formulation. Related quantities are the density at the nucleus and the electric field gradient. By removing a core electron, core-level binding energies can be directly calculated as a total-energy difference.

As main disadvantage, the complexity of the approach is to be emphasised. In addition to the larger effort of implementation, two major consequences are as follows:

- When applying a method using the LMTO basis set, a considerable number of parameters must be chosen sensibly. This starts with the basic partitioning of space when the atomic-sphere radii are defined and the choice of the basis set. After that, a daunting number of convergence parameters (such as angular-momentum cutoffs) must be specified.
- Modifications which should in principle be straightforward can be extremely difficult to do. As an example, consider the evaluation of the optical matrix elements, that is, the expectation value of the gradient operator $i\nabla$ between two wave functions. In a pure plane-wave basis set, this can be done in a few lines of code. In the LMTO basis set, this task is a major programming project.

Chapter 3

Description of the Method

The program under discussion here implements a variant of the LMTO method. It retains many basic features such as the partition of space into muffin-tin spheres and an interstitial region, atom-centered basis functions of a well-defined angular momentum, and augmentation. Two new ingredients are introduced:

- The first new element is to remove the singularities from the Hankel function envelopes, modifying the functions in the central sphere and the adjoining interstitial region. This lets the user tailor the basis functions in two ways: by specifying the decay at large radii, and by adjusting how much each function "bends over" as it approaches the central site. The extra freedom leads to a smaller basis. As a further advantage, suitable functions generally turn out to be smoother than the standard LMTO envelopes, permitting a reasonably efficient approach by numerical integration.

- The augmentation process is reformulated in a way which is closer to the pseudopotential formalism. The essence is that a quantity is represented by a smooth function extending through the whole cell, which is then modified by adding local terms inside each atomic sphere. This approach leads to smaller angular momentum cutoffs, independence of the sphere terms from the environment, and a simple force theorem.

These two elements are described below, followed by a section about the way in which these ideas are combined in a practical implementation.

3.1 Smooth Hankel Functions

Starting here, the advantages of a modified basis inspired by the LMTO functions are discussed. As a reminder, an LMTO envelope of negative

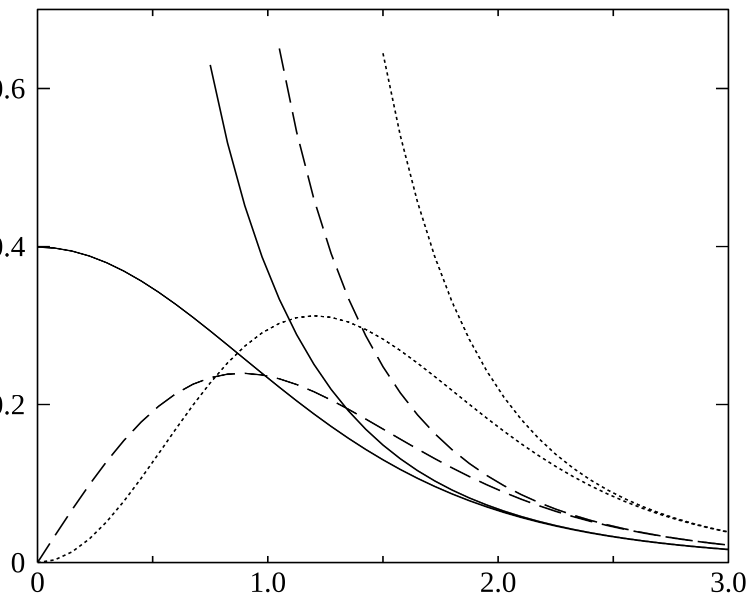

Figure 3.1: Comparison of smooth and standard Hankel functions for l=0 (continuous lines), l=1 (dashed), and l=2 (dotted lines). The energy ϵ equals -1 and the smoothing radius $R_{\rm sm}$ equals 1.0. For large radii, the smooth and standard functions coincide. Near the origin, the smooth function bends over gradually until it enters as r^l whereas the standard function has a singularity proportional to $1/r^{l+1}$.

energy solves the Schrödinger equation for a flat potential, decays exponentially at large distances, and has a singularity at the site where it is centered. The essence of the modification is to remove the singularity. The resulting "smooth Hankel functions" are smooth and analytic everywhere. For use as envelope functions, the parameters are chosen so that the functions bend away from the unsmoothed variants already outside the central atomic sphere. This speeds up the calculation for two separate reasons: the basis can be smaller, and numerical integration can be done using a considerably coarser mesh.

3.1.1 Basic Properties

In the context of setting up or running a calculation, the relevant information concerning the smooth Hankel functions [9, 10] can be taken from Fig. 3.1. For large radii, the smooth function to each angular momentum equals the corresponding standard Hankel function, showing the same exponential decay proportional to $\exp(-\kappa r)$, specified by the negative energy parameter $\epsilon = -\kappa^2$. At smaller radii, the function bends over gradually until it ultimately approaches r^l near $r = 0$. When multiplied by the spheric harmonic $Y_L(\hat{r})$, the result is analytic in all parts of space.

Of some importance is R_{sm}, the "smoothing radius" associated with the function. It turns out that the standard Hankel function and its smooth variant are (for practical purposed) equal where the gaussian $\exp(-r^2/R_{\text{sm}}^2)$ is negligible, say for $r > 3R_{\text{sm}}$. When R_{sm} is increased, the bending-over starts at larger values of r and the resulting function has been smoothed more strongly. Specifically, the values near $r = 0$ becomes smaller as the former singularity is washed out more and more.

Overall, two distinct parameters determine the shape of each function. The energy gives the decay at large radii, and the smoothing radius determines how strongly the function has been smoothed. To optimize the basis for a given type of atom, both parameters should be adjusted.

As a basis set, these functions combine many of the advantages of Hankel functions and gaussians. Thanks to the exponential wave-function-type behavior at large r, calculations using them are more stable than those using gaussians. Near the origin, they have a smooth nonsingular shape. Many important quantities can be evaulated analytically for these functions.

3.1.2 Formal Definition

For the interested reader, the smooth Hankel functions are constructed in the following way. The usual Hankel function for angular momentum zero is $h_0(r) = e^{-\kappa r}/r$ where κ defines the decay at large radii. As a function of $r = |\mathbf{r}|$ in three-dimensional space, h_0 satisfies the differential equation

$$(\Delta + \epsilon)h_0(r) = -4\pi\delta(\mathbf{r}) \tag{3.1}$$

where $\epsilon = -\kappa^2$ is the energy associated with the function, here always taken to be negative. Thus, $\Delta + \epsilon$ applied to h_0 is zero everywhere except at $\mathbf{r} = 0$, where there is a $1/r$ singularity. Expressed differently, $h_0(r)$ is the response of the operator $\Delta + \epsilon$ to a specific source term, namely a delta function.

To change this standard Hankel function into a smooth Hankel function, the infinitely sharp delta function is smeared out into a gaussian:

$$(\Delta + \epsilon)h_0(r) = -4\pi g_o(r) \ . \tag{3.2}$$

Given a suitable normalization of $g_0(r) = C\exp(-r^2/R_{\text{sm}}^2)$, the smooth Hankel approaches the standard function for large r. As r becomes smaller and reaches the range where $g_0(r)$ is non-negligible, the function bends over smoothly and behaves as a constant times r^l for $r \to 0$.

We also need smooth Hankel functions for higher angular momenta in order to construct basis functions for the s, p, d... states. In brief, these can be obtained immediately by applying a differential operator $\mathcal{Y}_L(-\nabla)$, defined as follows. The spheric harmonic polynomial $\mathcal{Y}(\mathbf{r}) = r^l Y_L$ is a polynomial in x, y, and z, for example $C(x^2 - y^2)$. By substituting the partial derivatives $-\partial_x$, $-\partial_y$, and $-\partial_z$ for x, y, and z, respectively, the required operator is obtained in a straightforward manner. Applying this operator to the delta function yields point dipoles, quadrupoles and so on, and applying it to $g_0(r)$ yields smeared-out gaussian versions of these. Thus the L-th smooth Hankel functions is $H_L(\mathbf{r}) = \mathcal{Y}_L(-\nabla)h_0(r)$ and satisfies the differential equation

$$(\Delta + \epsilon)H_L = -4\pi G_L(\mathbf{r}) = -4\pi\mathcal{Y}_L(-\nabla)g_0(r)\ . \tag{3.3}$$

Several important quantities can be calculated analytically for these functions, for example the overlap integral and the kinetic energy expectation value between two of them. They can also be expanded around some point in the unit cell. For further details, see Ref. [10].

3.1.3 Reducing the Basis Size

The first reason for using the smooth-Hankel basis functions is that this can reduce the size of the basis set, leading to a substantial gain in efficiency. To show this, note that the standard LMTO basis functions are in fact not optimal as a basis for representing the crystal or molecular wave functions. The main problem is that they are "too steep" in the interstitial region close to the muffin-tin sphereon which they are centered. This is illustrated in Fig. 3.2. The standard Hankel functions solves Schrödinger's equation for a flat potential. When approaching a nucleus, the true crystal potential is not flat but decreases as it feels the attractive nucleus. The curvature of the wavefunction is the potential minus the energy which therefore becomes negative. The wavefunction therefore bends over already outside the MT sphere. By using smooth Hankel functions, this typical form is inherent in each basis function.

When expanding the crystal wavefunction using standard Hankel functions, generally the basis set must include some slowly decaying functions together with others which are considerably more localized. In the course of the calculation these combine with opposite signs, in this way modeling the required bending-over. When the envelopes already have the correct behavior, some of the additional localised functions can be left away.

In practice, the amount of gain depends on the type of atom. For the important angular momenta, a tripled basis can often be replaced by a

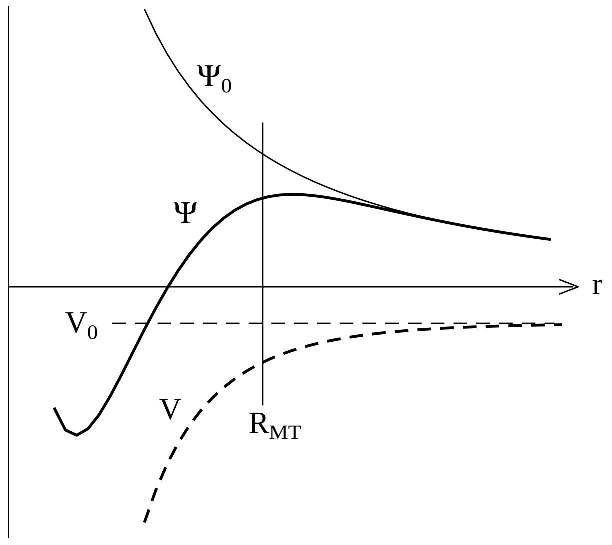

Figure 3.2: Sketch to explain why smooth Hankel functions lead to an improved basis. For the flat potential V_0, the solution of the radial Schrödinger equation Ψ_0 is a standard Hankel function with a singularity at the origin. As the true potential V starts to feel the attractive nuclear potential, the correct wavefunction Ψ bends over. This behavior already starts outside the muffin-tin radius and is built into the smooth Hankel functions.

doubled set. Less important channels such as the *d* states in an *sp* atom can be described by one radial function instead of two. An overall reduction by a factor of almost two is possible. In the order(N^3) steps, the computer time in such an optimal case divides by eight.

3.1.4 Reducing the Numerical Effort

For a full-potential calculation, the smooth Hankel functions help to reduce the numerical effort. A full-potential calculation is one which does not impose any shape approximation on the potential or the density. One common approximation which is not full-potential is the muffin-tin form, consisting of spherically averaged potentials in the spheres and a flat interstitial potential. This, or the related atomic sphere approximation (ASA), can produce accurate energy bands, but not the small energy differences when atoms are shifted about. To go beyond the muffin-tin and ASA approximations, a completely general potential shape should be permitted inside the atomic spheres and in the interstitial region. Inside the spheres, this improvement is straightforward, albeit somewhat tedious to implement. It is the interstitial region which poses a major problem.

The Interstitial Potential Problem

With only slight exaggeration, for a full-potential calculation using the LMTO basis set, there is only *one* major problem to be solved. This task is to evaluate the matrix elements of the interstitial potential matrix elements for the chosen basis. Thus, if $H_i(\mathbf{r})$ and $H_j(\mathbf{r})$ are two envelope functions, the problematic term is the integral

$$V_{ij}^{(\mathrm{IR})} = \int_{\mathrm{IR}} H_i^*(\mathbf{r}) V(\mathbf{r}) H_j(\mathbf{r})\, d\mathbf{r}\ . \tag{3.4}$$

Here i and j are compound indices which specify the basis functions in terms of position, angular momentum, and localisation, IR denotes the interstitial region, and the star indicates complex conjugation. This matrix element always causes problems, no matter what representation is chosen for the interstitial potential. In fact, a major part of devising a viable method is to select a representation for the density and the potential which makes it possible to perform this step reasonably efficiently.

Basically there are two ways to proceed. First, the potential in the interstitial region can also be expanded in some suitable set of functions. When done cleverly, this can lead to a very compact representation. Substituting the expansion under the integral in (3.4) leads to a sum of integrals, each over a product of three terms. Unfortunately, such integrals can almost never be evaluated in closed form (the notable exception is when all terms are gaussians). In our context, two of the terms are LMTO envelopes and then no reasonable choice of the potential expansion leads to a closed form.

Alternatively, the potential can be specified by tabulating it on a regular mesh which extends through the unit cell. Equivalently, the coefficients of the Fourier expansion can be given. The immediate price to be paid is that more data is needed using such a numerical representation. However, there are also some considerable advantages. Since the exchange-correlation potential must be evaluated point by point, a mesh representation is needed somewhere along the way in any case. By using a regular mesh, the Poisson equation can be solved easily using a fast Fourier transform. The unsavory step of fitting the output density or the effective potential to a set of auxillary functions is avoided. Things like gradient corrections can be implemented much more easily on a regular mesh than in a more complicated representation.

Potential Integrals for Smooth Hankels

When a regular mesh is used, the best way to evaluate the difficult interstitial potential matrix element (3.4) is by integrating over the whole unit cell and subtracting the parts from the spheres. The envelope functions are extended in a smooth but as yet unspecified manner through the MT spheres. The interstitial potential is also extended in a similar way. For these smooth functions, the potential integral is evaluated over the whole unit cell by summing over the regular mesh. After that, the contributions from inside the MT spheres are subtracted, leaving the integral over the IR. This last step can be combined with the augmentation itself, which adds the corresponding contributions involving the true numerical basis functions and potential.

The problem with calculating three-dimensional integrals using a mesh is that the associated computational effort soon swamps everything else. To keep the effort manageable, it is of high priority to make the integrands as smooth as possible. This can be done by using smooth Hankels as envelopes. As an example, consider silicon with a muffin-tin radius of 2.2 bohr. For the standard LMTO basis, the smoothing must be noticeable inside the MT sphere only, demanding a smoothing radius no larger than 0.6 to 0.7 bohr. Outside the central sphere, the smooth and unsmooth Hankel functions are then identical to acceptable precision. The required integration mesh spacing is about 0.35 bohr. If we permit the functions to bend over outside the MT sphere, we find that the optimal basis functions have a smoothing radius of about 1.4 bohr. For these functions, the integration mesh can be twice as coarse. Consequently the number of mesh points and the computational effort are divided by eight.

Altogether, a modified basis using “smooth Hankel functions” combines two major advantages. Since these are more similar to the final wavefunctions, adequate convergence can be attained using a smaller basis set. Secondly, since each function is smoother, the mesh used to evaluate the potential integral can be coarser. These effects combine to a substantial saving in computer time. As an estimate (admittedly for an extremely favorable case), the integration mesh can be twice as coarse, leading to a saving of about $(1/2)^3 = 1/8$ in the order-3 steps. The basis set will contain something like one-half of the functions needed without extra smoothing, giving another factor of approximately 1/8. Together, the computer time is divided by 64. While the overall gain will not be as large for many systems, a speedup by a factor between 10 to 20 seems realistic.

3.1.5 Expansion Around a Site

One of the steps which is needed in an implementation is to expand a smooth Hankel function around some point in the unit cell, usually an atomic site. Far away from the center, the smooth Hankel function equals the unsmooth variant and the well-known structure constant expansion for the standard Hankel functions could be used. On the central sphere, the function is given explicitly by its definition. It is for sites close to the central sphere, such as nearest-neighbor atoms, where something new is needed. Here the function generally starts to bend over and the standard expansion does not apply.

This problem is solved as follows. We define a family of higher-order gaussians $G_{pL}(\mathbf{r})$ by applying differential operators to the seed function $g_0(r) = C\exp(-r^2/R_{\rm sm}^2)$:

$$G_{pL}(\mathbf{r}) = \Delta^p \mathcal{Y}_L(-\nabla) g_0(r)\ . \tag{3.5}$$

We can construct biorthogonal polynomials to these functions, *i.e.*, a set of polynomials $P_{pL}(\mathbf{r})$ with the property

$$\int G_{pL}(\mathbf{r}) P_{p'L'}(\mathbf{r})\, d\mathbf{r} = \delta_{pp'}\delta_{LL'}\ . \tag{3.6}$$

In fact, P_{pL} is just G_{pL} divided by $g_0(r)$, suitably normalized. To expand an arbitrary function $f(\mathbf{r})$ as a sum of the P_{kL}, each coefficient is simply the integral over $f(\mathbf{r})$ times the corresponding gaussian:

$$f(\mathbf{r}) = \sum_{pL} A_{pL} P_{pL}(\mathbf{r}) \tag{3.7}$$

where

$$A_{p_L} = \int f(\mathbf{r}) G_{pL}(\mathbf{r})\, d\mathbf{r}\ . \tag{3.8}$$

This expansion, when truncated to some low value of p, is considerably more accurate than, for example, a Taylor series. This is because the expansion converges smoothly towards $f(\mathbf{r})$ in the range where $g_0(r)$ is large as more terms are included. When $f(\mathbf{r})$ is a smooth Hankel function centered anywhere in space, the integral can be done analytically. This supplies the desired local expansion.

The expansion is used in several different steps, most prominently to augment the envelope functions. The important thing is to understand the significance of the two parameters which influence the accuracy of the expansion. By chosing a cutoff $p_{\rm max}$ for the terms in the expansion, the radial function is represented as a polynomial of order $p_{\rm max}$. The range over which the expansion is usable is determined by the smoothing radius $R_{\rm sm}$ of the gaussians G_{pL}. When $R_{\rm sm}$ is chosen larger, the expansion can be used over a larger part of space but will not be as accurate overall for the same value of $p_{\rm max}$. Taking $R_{\rm sm}$ close to one-third of the muffin-tin radius will usually give a reasonable expansion within the muffin-tin sphere.

3.2 New Formulation of Augmentation

This section describes the second major ingredient, a reformulation of the way in which augmentation is done. In large part, this is an attempt to squeeze the formalism of augmentation into a form which is similar to the pseudopotential approach.

Why should this be desirable? Pseuodopotential methods are rightly popular thanks to their conceptual and practical simplicity. They also treat the region near the nucleus separately, but this is done using a simple additive, spherical, and environment-independent (but l-dependent) potential. Angular momentum cutoffs are generally very low and an expression for the force is easy to obtain. In contrast, augmentation works by cutting out the part of space near the nucleus and expanding quantities there to relatively high angular momenta. Things are so complicated that it is often difficult to find a valid force theorem.

In practice, the augmentation and pseudopotential approaches turn out to have some similarity. Both expand a set of smooth basis functions by angular momentum around the different sites, then operate on the different l components independently. This also suggests that a more unified description should be possible.

A special point of interest is the question of angular-momentum cutoffs. Norm-conserving pseudopotentials [11], which generally work well, are carefully constructed to contain the right charge in each angular-momentum component. However, they are not tuned to reproduce the non-spherical density. For example, the p component of the density is assembled from the product of the s and p partial waves and similar terms. There is no obvious reason why these should be reproduced properly by the pseudowavefunction. The pseudopotential method has managed to push a substantial portion of the problem onto the smooth density. The aim is to do the same in the all-electron context. The approach presented here contains some of the elements of Vanderbilt's ultrasmooth pseudopotentials [12] and the projector augmented wave method [13]. Going beyond these, it gives a complete and generally applicable formulation for all-electron methods which has the simplicity and tranferability of the pseudopotential approach.

3.2.1 Smooth and Local Terms

In the method presented here, some effort was taken to formulate matters such that low angular-momentum cutoffs can be used. Quantities defined throughout the crystal are consistently represented as a smooth function, extending throughout the whole unit cell, plus contributions from inside the muffin-tin spheres. This is not the same as a piecewise definition, which would *replace* the smooth function by something else inside the spheres. The definition used here is the key to good accuracy at low angular-momentum cutoffs, since it permits the smooth interstitial function to supply the higher angular momentum components.

For example, to represent the valence density we start with a smooth function $n_0(\mathbf{r})$ tabulated on the real-space mesh. For each atom ν, we carry about a "true" density, which is to be added on, and a "smooth" contribution, which is to be subtracted and should be a reasonable approximation to the local expansion of $n_0(\mathbf{r})$ of the chosen angular-momentum cutoff:

$$n(\mathbf{r}) = n_o(\mathbf{r}) + \sum_\nu \Big\{ \rho_{1\nu}(\mathbf{r}) - \rho_{2\nu}(\mathbf{r}) \Big\} \ . \tag{3.9}$$

As will be seen later on, $\rho_{1\nu}$ includes the core and the nucleus, and $\rho_{2\nu}$ includes their smooth "pseudo" versions, modeled by localized gaussians. The local contribution $\rho_\nu = \rho_{1\nu} - \rho_{2\nu}$ is a sum over various angular momentum components, is non-zero only inside the corresponding MT sphere, and goes to zero smoothly as it approaches the MT radius R_{mt}. The important feature is that even if ρ_ν is truncated to a low angular momentum, possibly only to its spherical part, the total density $n(\mathbf{r})$ includes contributions to for all L up to infinity. At high angular momentum the radial part for any smooth function starts to look like a constant times r to the lth power. It is pointless to include these components explicitly in the local representation, since they are already contained in $n_0(\mathbf{r})$.

Basically, the crystal potential is treated in the same way:

$$V(\mathbf{r}) = \tilde{V}_0(\mathbf{r}) + \sum_\nu \Big\{ V_{1\nu}(\mathbf{r}) - \tilde{V}_{2\nu}(\mathbf{r}) \Big\} \tag{3.10}$$

where the separate terms play the same role as for the density. The potential is made by solving the Poisson equation for the input density and adding the exchange-correlation potential. In the first of these steps, a smooth "compensated" density is made which has the correct multipole moments in the spheres. The tilde over a term indicates that the compensating charges enter into its definition, as will be described next.

3.2.2 Making the Effective Potential

At the start of each iteration, the crystal density is available in the form given in (3.9) above. The first step is to make the effective potential and to evaluate the associated energy terms.

Electrostatic Energy and Potential

To make the electrostatic potential, a smooth "pseudodensity" is constructed which equals the correct density in the interstitial region and has the correct multipole moments in all the spheres. This is done by adding localized gaussians g_ν to the smooth mesh density which have the same moments as the local contributions $\rho_{1\nu} - \rho_{2\nu}$ within the spheres:

$$\tilde{n}_0 = n_0 + \sum_\nu g_\nu \tag{3.11}$$

The potential of these gaussians is of the same form as the smoothed nuclear potential and is handled together with this in reciprocal space. The electrostatic potential of the smooth mesh density $n_0(\mathbf{r})$ is made by transforming to reciprocal space using a fast Fourier transform, dividing the coefficients by the squares of the reciprocal vectors, and transforming back. Together, this produces an electrostatic potential $\tilde{V}_0^{\mathrm{es}}(\mathbf{r})$ which is valid throughout the interstitial region and extends smoothly through the spheres.

Next, the electrostatic potential inside the spheres is fixed up. To recover the true density from the pseudodensity, the following quantity must be added at each site ν:

$$\tilde{\rho}_\nu = \rho_{1\nu} - \rho_{2\nu} - g_\nu \ . \tag{3.12}$$

By construction, this local contribution has multipole moments which are zero. We solve the Poisson equation twice to obtain the "true" local potential $V_{1\nu}^{\mathrm{es}}$ and the "smooth" local potential $V_{2\nu}^{\mathrm{es}}$

$$\Delta V_{1\nu}^{\mathrm{es}} = -8\pi\rho_{1\nu} \tag{3.13}$$

$$\Delta V_{2\nu}^{\mathrm{es}} = -8\pi(\rho_{2\nu} + g_\nu) \ . \tag{3.14}$$

The source terms $\rho_{1\nu}$ and $\rho_{2\nu} + g_\nu$ have the same multipole moments, and we are really only interested in the difference $V_{1\nu}^{\mathrm{es}} - V_{2\nu}^{\mathrm{es}}$. Thus, the boundary conditions when solving the Poisson equation are irrelevant, as long as the same set is used in both equations, and can be set to zero.

Finally, the electrostatic energy is obtained by adding together the smooth term

$$\int \tilde{n}_0(\mathbf{r})\tilde{V}_0^{\mathrm{es}}(\mathbf{r})\, d\mathbf{r} \tag{3.15}$$

and the local terms

$$\int_{S_\nu} \rho_{1\nu} V_{1\nu}^{\mathrm{es}}\, d\mathbf{r} - \int_{S_\nu} (\rho_{2\nu} + g_\nu) V_{2\nu}^{\mathrm{es}}\, d\mathbf{r} \tag{3.16}$$

for all sites.

General Note on Energy Integrals

The expression (3.15) plus (3.16) for the electrostatic energy looks natural enough, but there is an important subtlety in there. If Equation (3.9) for the density is taken seriously, why not simply add together $\rho_{1\nu}$ and $-\rho_{2\nu}$ for each site from the very beginning? Instead, the program explicitly carries about both functions in separate arrays. The sum of (3.15) and (3.16) is not, in fact, the exact electrostatic energy for the density in (3.9); this would involve a large number of cross terms between n_0, $\rho_{1\nu}$, and $\rho_{2\nu}$.

The point is that we have a "true" and a "smooth" part to the problem which should be kept separate. When evaluating energy integrals like the one above, the true density should not interact with the smooth potential and vice versa. Formally, we consider the crystal density as given by a triplet $(n_0, \{\rho_{1\nu}\}, \{\rho_{2\nu}\})$ whereby each component exists in a separate space. The various manipulations (making potentials, evaluating energy terms etc.) are done in each space independently, and the results are added together to get the final energy integrals.

It is worth emphasising that this somewhat mysterious business is only relevant for the *rate* of convergence with the angular-momentum cutoff, and not with the final outcome at convergence. When all angular-momentum components up to infinity are included, the distinction between the different ways to calculate the integral disappears. In the end, the two main consequences of the formulation are:

- In the resulting energy integrals, the higher angular momenta are supplied by the smooth density $n_0(\mathbf{r})$. That is, we start with an expression (3.15) which contains angular momentum components up to infinity in each sphere, then replace a few of the lower ones by adding (3.16).
- The augmentation procedure, described further on, cleanly separates into a smooth part and into a local contribution, whereby the later is independent of the environment of the atom.

Exchange-Correlation Energy and Potential

The exchange-correlation energy is calculated by integrating a smooth function over the whole unit cell, then replacing the smooth contribution by the true one inside each MT sphere. Things are simpler here than in the context of the electrostatic potential because no additional terms are needed to fix up the multipole moments. Thus, the smooth exchange-correlation potential is made by evaluating it point-by-point for the smooth mesh density $n_0(\mathbf{r})$. Adding this to the electrostatic potential for the compensated density gives the total mesh potential

$$\tilde{V}_0(\mathbf{r}) = \tilde{V}_0^{\mathrm{es}}(\mathbf{r}) + \mu_{\mathrm{xc}}(n_0(\mathbf{r})) \tag{3.17}$$

Inside each sphere, two different exchange-correlation potentials are made and added respectively to the true and smooth local electrostatic potentials:

$$\begin{aligned} V_{1\nu}(\mathbf{r}) &= V_{1\nu}^{\mathrm{es}}(\mathbf{r}) + \mu_{\mathrm{xc}}(\rho_{1\nu}(\mathbf{r})) && (3.18) \\ V_{2\nu}(\mathbf{r}) &= V_{2\nu}^{\mathrm{es}}(\mathbf{r}) + \mu_{\mathrm{xc}}(\rho_{2\nu}(\mathbf{r})) && (3.19) \end{aligned}$$

to produce the final potentials seen by the true and smooth densities at this site. The XC energy is made by constructing the energy density ϵ_{xc} in exactly the same way and adding together the smooth mesh term

$$\int n_0 \epsilon_{\mathrm{xc}}(n_0)\, d\mathbf{r} \tag{3.20}$$

and the local contributions

$$\int_{S_\nu} \rho_{1\nu}\epsilon_{\mathrm{xc}}(\rho_{1\nu})\, d\mathbf{r} - \int_{S_\nu} \rho_{2\nu}\epsilon_{\mathrm{xc}}(\rho_{2\nu})\, d\mathbf{r}\ . \tag{3.21}$$

Since $\epsilon_{\mathrm{xc}}(\rho)$ is a nonlinear function of the density, it mixes together all angular momentum components up to infinity. That is, $\epsilon_{\mathrm{xc}}(\rho_L+\rho_K)$ is not the same as $\epsilon_{\mathrm{xc}}(\rho_L)+\epsilon_{\mathrm{xc}}(\rho_K)$ and we should really include all L terms, even when making something as fundamental as the spherical part of $\epsilon_{\mathrm{xc}}(\rho)$. By evaluating the exchange-correlation energy as described, the interaction with the higher angular momenta is taken over by the smooth mesh contribution. When $\rho_{1\nu}$ and $\rho_{2\nu}$ are truncated to the same low angular momentum, the errors in the two integrands $\rho_{1\nu}\epsilon_{\mathrm{xc}}(\rho_{1\nu})$ and $\rho_{2\nu}\epsilon_{\mathrm{xc}}(\rho_{2\nu})$ are similar. Thus, while the two integrals in (3.21) converge rather sedately with the L cutoff separately, their difference converges rapidly.

3.2.3 Hamiltonian and Overlap

At this stage, the effective potential in the crystal and the corresponding total-energy terms have been made. The next step is to calculate the Hamiltonian and overlap matrix elements for some set of basis functions $\chi_i(\mathbf{r})$ used to represent the crystal wavefunction:

$$\begin{aligned} H_{ij} &= \int \chi_i^*(\mathbf{r})[-\Delta + V_{\mathrm{eff}}(\mathbf{r})]\chi_j(\mathbf{r})\, d\mathbf{r} && (3.22) \\ S_{ij} &= \int \chi_i^*(\mathbf{r})\chi_j(\mathbf{r})\, d\mathbf{r}\ . && (3.23) \end{aligned}$$

These are integrals over the unit cell, which will be evaluated using the same considerations as for the total-energy terms in the previous section.

The central element here is that the basis functions are made by augmenting some smooth "envelope functions." These might be smooth Hankel functions, plane waves, or something else. Augmentation means, roughly, to replace each function inside every muffin-tin sphere by a numerical solution of the radial Schrödinger equation which matches smoothly at the sphere boundary. Starting from some unspecified set of envelope functions $\{F_i(\mathbf{r})\}$, the first step is to project out local information about the envelopes near the chosen site. This requires some local set of radial functions, call them P_{kL}, used to expand the i-th envelope as

$$F_i(\mathbf{r}) = \sum_{kL} C_{kL}^{(i)} P_{kL}(\mathbf{r}) \ . \tag{3.24}$$

The notation anticipates that we will later use the polynomials of Section 3.1.5, but at this stage the P_{kL} are general functions with well-defined angular momentum and sufficient degrees of freedom.

To augment, we first construct functions $\tilde{P}_{kL}$ which are augmented versions of the P_{kL}. That is, assuming P_{kL} is given in the form

$$P_{kL}(\mathbf{r}) = p_{kl}(r) Y_L(\hat{r}) \tag{3.25}$$

then the augmented version is

$$\tilde{P}_{kL}(\mathbf{r}) = \tilde{p}_{kl}(r) Y_L(\hat{r}) = \Big[A_{kl}\phi_l(r) + B_{kl}\dot{\phi}_l(r) \Big] Y_L(\hat{r}) \tag{3.26}$$

where $\tilde{p}_{kl}(r)$ equals the contents of the square brackets. In a standard way, $\phi_l(r)$ and $\dot{\phi}_l(r)$ are a specific solution of the radial Schrödinger equation and its energy derivative, respectively. The coefficients A_{kl} and B_{kl} are chosen so that $\tilde{p}_{kl}$ and p_{kl} have the same value and derivative at the muffin-tin radius R_{mt}. The augmented envelope function then is

$$\tilde{F}_i(\mathbf{r}) = F_i(\mathbf{r}) + \sum_{kL} C_{kL}^{(i)} \Big\{ \tilde{P}_{kL}(\mathbf{r}) - P_{kL}(\mathbf{r}) \Big\} \ . \tag{3.27}$$

At first sight this expression seems nonsensically complicated. Using (3.24), the first and last terms should cancel, leaving only a sum over the $\tilde{P}_{kL}$. The point is that when the sums over k and L are truncated, as will be assumed from here on, we still obtain something very similar to the untruncated sum. With the same logic as before, we start with a function containing all components up to infinity and replace only a few of the lower terms by numerical functions.

Overlap integral

Our aim is to calculate the Hamiltonian and overlap integrals for the augmented envelope functions. The overlap between two such functions is calculated as follows:

$$\int \tilde{F}_i^* \tilde{F}_j \, d\mathbf{r} = \int F_i^* F_j \, d\mathbf{r} + \sum_{kk'L} C_{kL}^{(i)*} \sigma_{kk'l} C_{k'L}^{(j)} \tag{3.28}$$

where

$$\sigma_{kk'l} = \int_S \Big\{ \tilde{P}_{kL} \tilde{P}_{k'L} - P_{kL} P_{k'L} \Big\} \, d\mathbf{r} \ . \tag{3.29}$$

For simplicity, it was assumed here that there is one atom in the unit cell and that P_{kL} is real. Note also that $\sigma_{kk'l}$ depends only on l, not on L.

Three comments should be made about the expression for the overlap integral. First of all, this is a thoroughly appealing representation. We start by calculating the overlap between the smooth unaugmented envelopes. Then we project out local information about the envelopes in the form of the vectors $C^{(i)}$ and $C^{(j)}$. To augment, we add on the product $C^{(i)\dagger} \sigma C^{(j)}$ where σ is a small symmetric matrix characterizing the atom at this site.

Secondly, we can repeat the discussion of the convergence with the expansion cutoffs presented previously for the total-energy integrals. Equation (3.28) is not strictly equal to the integral over $\tilde{F}_i^* \tilde{F}_j$ when these functions are given by (3.27). Doing this integral exactly would involve a large number of unwieldy cross terms. For finite cutoffs, (3.28) is a good approximation to the result which would be obtained if no truncation were done. This is because the truncation errors are similar in the integrals over $\tilde{P}_{kL} \tilde{P}_{k'L}$ and $P_{kL} P_{k'L}$ and these therefore largely cancel when the the difference is taken.

Finally, assume for a moment that the radial augmentation functions ϕ_l and $\dot{\phi}_l$ are kept frozen throughout the calculation, which is not a bad procedure in practice. Then the $\tilde{P}_{kL}$ are also fixed and the local matrix $\sigma_{kk'l}$ is completely independent of the environment. The formulation is thus begins to approach that of a unique, transferable pseudopotential.

Kinetic energy integral

Given these preparations, we can breeze through the next term, the kinetic energy integral. This is calculated as

$$\int \tilde{F}_i^* [-\Delta] \tilde{F}_j \, d\mathbf{r} = \int F_i^* [-\Delta] F_j \, d\mathbf{r} + \sum_{kk'L} C_{kL}^{(i)*} \tau_{kk'l} C_{k'L}^{(j)} \tag{3.30}$$

where

$$\tau_{kk'l} = \int_S \Big\{ \tilde{P}_{kL}[-\Delta]\tilde{P}_{k'L} - P_{kL}[-\Delta]P_{k'L} \Big\}\, d\mathbf{r} \ . \tag{3.31}$$

The only noteworthy feature is that the local kinetic energy matrix $\tau_{kk'l}$ is symmetric. When the operator $-\Delta$ is moved from the second to the first function under each integral, two surface terms arise. These cancel because $\tilde{P}_{kL}$ and P_{kL} match to value and slope. As before, $\tau_{kk'l}$ is independent of the environment if the radial augmentation functions ϕ_l and $\dot{\phi}_l$ are kept frozen.

Potential energy integral

Finally, we consider the potential matrix element, which is somewhat more complicated. The electrostatic potential inside a given sphere depends not only on the density inside it, but also on the density in all other parts of the unit cell. It is not obvious how a transferable local potential can be separated from the overall problem, although this is the fundamental assumption of the pseudopotential method.

In precise terms, the potential is an auxillary function needed to minimize the total energy respective to the trial density. It should give the first-order response of the electrostatic and exchange-correlation energy for a variation of the density. When evaluating the electrostatic energy, compensating gaussians were added to the smooth mesh density to make a "pseudodensity" with the correct multipole moments in the spheres. This modified the energy and will therefore lead to some additional terms in the potential matrix elements.

As a reminder, if the local sphere density $\rho_{1\nu}-\rho_{2\nu}$ has multipole moments q_M, the compensated mesh density is

$$\tilde{n}_0(\mathbf{r}) = n_0(\mathbf{r}) + \sum_M q_M G_M(\mathbf{r}) \tag{3.32}$$

where G_M is a gaussian of unity moment to angular momentum M, localized inside the muffin-tin sphere except for a negligible tail.

As before, the potential energy integral is in the form

$$\int \tilde{F}_i^* V \tilde{F}_j \, d\mathbf{r} = \int F_i^* \tilde{V}_0 F_j \, d\mathbf{r} + \sum_{kk'LL'} C_{kL}^{(i)*} \pi_{kk'LL'} C_{k'L'}^{(j)} \ . \tag{3.33}$$

One difference to the two previous cases is that the local potential matrix $\pi_{kk'LL'}$ is not diagonal in L unless the local potentials are taken as spherical. To derive an expression for $\pi_{kk'LL'}$ we look at the energy change for a variation of the density. Without going into the details, the result involves the multipole moments of $\tilde{P}_{kL}\tilde{P}_{k'L'} - P_{kL}P_{k'L'}$:

$$Q_{kk'LL'M} = \int_S \Big\{ \tilde{P}_{kL}\tilde{P}_{k'L'} - P_{kL}P_{k'L'} \Big\}\, r^m Y_M(\hat{r})\, d\mathbf{r} \ . \tag{3.34}$$

The matrix $\pi_{kk'LL'}$ turns out to be the sum of two contributions. The first involves the smooth potential $\tilde{V}_0$ for the compensated mesh density:

$$\pi^{\text{mesh}}_{kk'LL'} = \sum_M Q_{kk'LL'M} \int \tilde{V}_0 G_M \, d\mathbf{r} \tag{3.35}$$

and second involves the true and the smooth local potentials, V_1 respectively $\tilde{V}_2$:

$$\begin{aligned}\pi^{\text{local}}_{kk'LL'} &= \int_S \left\{ \tilde{P}_{kL} V_1 \tilde{P}_{k'L'} - P_{kL} \tilde{V}_2 P_{k'L'} \right\} d\mathbf{r} \\ &- \sum_M Q_{kk'LL'M} \int_S V_2 G_M \, d\mathbf{r} \; . \end{aligned} \tag{3.36}$$

As an unexpected bonus, these expressions have moved the situation significantly towards the desired environment-independence of the local term. In (3.36), the true potential is felt by the partial density $\tilde{P}_{kL}\tilde{P}_{k'L'}$ while the smooth potential is felt by $P_{kL}P_{k'L'} + \sum Q_{kk'LL'M}G_M$. These two quantities have the same multipole moments; this was essentially the definition of the $Q_{kk'LL'M}$. If we change the boundary conditions for the electrostatic potential in the sphere in any way, this just adds the same linear combination of the functions $r^m Y_M(\mathbf{r})$ to both V_1 and $\tilde{V}_2$. It follows that the result of (3.36) does not depend on these boundary conditions, and we can just set them to zero.

The result is that (3.36) defines a quantity which can be calculated completely and unambiguously using only the density inside the sphere. Does that automatically make it a fixed quantity when the $\tilde{P}_{kL}$ are kept frozen, as discussed above? Not quite, because the local sphere density $\rho_1 - \rho_2$ can change in the course of a calculation, which would lead to changes in V_1 and $\tilde{V}_2$. Keeping the local sphere density frozen is clearly an additional approximation, albeit plausible, which must be assumed in order to get full independence from the environment.

There is still the term $\pi^{\text{mesh}}_{kk'LL'}$ in (3.35) to be considered. Augmentation modifies the density inside the muffin-tin spheres. Hereby the charge and the higher multipole moments are changed. In a pseudopotential analog, the procedure is more similar to Vanderbilt's [12] ultrasmooth pseudopotential than to the norm-conserving formulation. In the smooth part of the problem, the change of the moments is mapped onto gaussians of the correct normalisation. The term in (3.35) adds on the interaction of these gaussians with the smooth mesh potential.

3.2.4 Making the Output Density

The final step needed to complete the self-consistency loop is to construct the output density for the next iteration:

$$n^{\text{out}}(\mathbf{r}) = \sum_n w_n \left|\psi_n(\mathbf{r})\right|^2 \tag{3.37}$$

where the sum runs over the occupied eigenstates and the w_n are occupation numbers, in the simplest case equal to two because of spin degeneracy. Each wavefunction is a linear combination of the basis functions in the form

$$\psi_n(\mathbf{r}) = \sum_i T_{in} \tilde{F}_{in}(\mathbf{r}) \tag{3.38}$$

and the output density is a sum over the products $\tilde{F}_i^* \tilde{F}_j$:

$$n^{\text{out}}(\mathbf{r}) = \sum_{ij} \Big\{ \sum_n w_n T_{in}^* T_{jn} \Big\} \tilde{F}_i^*(\mathbf{r}) \tilde{F}_j(\mathbf{r}) \; . \tag{3.39}$$

The output density must be in the same form as the input density in (3.9), that is, given by a smooth mesh density n_0^{out} together with separate true and smooth local contributions $\rho_{1\nu}^{\text{out}}$ and $\rho_{2\nu}^{\text{out}}$ for each sphere ν. Given the preceeding discussion of the augmentation process, it is clear how to do this. Again assuming one atom in the cell for simplicity, the product of two augmented basis functions should be calculated as

$$\tilde{F}_i^* \tilde{F}_j = F_i^* F_j + \sum_{kk'LL'} C_{kL}^{(i)*} \Big\{ \tilde{P}_{kL} \tilde{P}_{k'L'} - P_{kL} P_{k'L'} \Big\} C_{k'L'}^{(j)} \; . \tag{3.40}$$

The integral over this quantity gave the overlap matrix, and the integral over this quantity times the potential gave the potential matrix element. In the latter case, the first term interacted with the smooth mesh potential $\tilde{V}_0$, the second with the local true potential V_1, and the third with the local smooth potential V_2 in the sphere. Thus, the accumulated sums over the first, second, and third terms produce n_0^{out}, $\rho_{1\nu}^{\text{out}}$, and $\rho_{2\nu}^{\text{out}}$, respectively.

3.2.5 Force Theorem

The force on an atom is defined as the negative gradient of the total energy respective to the corresponding atomic coordinates:

$$\mathbf{F}_\mu = -\nabla_\mu E_{\text{SC}}(\mathbf{R}_1, \ldots, \mathbf{R}_\mu, \ldots, \mathbf{R}_N) \; . \tag{3.41}$$

Here E_{SC} is the full self-consistent total energy, which depends only on the atomic positions. In terms of small differences, the prescription to make the force is: "Make the system selfconsistent at the given geometry, shift atom μ by a small amount, make the system selfconsistent a second time, and compare these two total energies." A force theorem is a closed expression, making it possible to calculate the forces $\mathbf{F}_\mu$ without explicitly shifting the atoms. Since the self-consistency process mixes together all the different energy terms, a straightforward differentiation of the total energy expression is a very strenuous (and not always successful) way to obtain such a theorem.

A more convenient way to derive a force theorem is as follows [14]. Assume we want to calculate the forces at the geometry $\mathbf{P}^{(0)} = (\mathbf{R}_1^{(0)}, \ldots, \mathbf{R}_N^{(0)})$, for which the self-consistent density is known. For this purpose we make *any* reasonable guess at the way in which the density changes as the atoms are moved. That is, we define a density $\tilde{n}_\mathbf{P}(\mathbf{r})$ for each geometry $\mathbf{P} = (\mathbf{R}_1, \ldots, \mathbf{R}_N)$ which conserves the total charge and which approaches the correct self-consistent density smoothly as $\mathbf{P} \to \mathbf{P}^{(0)}$. From the variational properties of the energy functional it follows that

$$\nabla_\mu E_\mathrm{SC}(\mathbf{R}_1, \ldots, \mathbf{R}_N) = \nabla_\mu \tilde{E}(\mathbf{R}_1, \ldots, \mathbf{R}_N) \qquad (3.42)$$

where $\tilde{E}$ is defined as the Harris energy at the guessed density:

$$\tilde{E}(\mathbf{R}_1, \ldots, \mathbf{R}_N) = E_\mathrm{H}[\tilde{n}_\mathbf{P}(\mathbf{r})] \; . \qquad (3.43)$$

This means that the forces can be obtained just as well by differentiating the auxillary function $\tilde{E}$. This is a much easier task than differentiating the self-consistent energy. Furthermore, since the guessed function $\tilde{n}_\mathbf{P}(\mathbf{r})$ can be chosen freely, different force theorems can be obtained depending on this choice. Sensibly, $\tilde{n}_\mathbf{P}(\mathbf{r})$ should be defined in a way which makes it easy to do the differentiation of $\tilde{E}$.

In the present formalism, the charge density is defined by a smooth mesh density $n_0(\mathbf{r})$ together with true and smooth local terms $\rho_{1\nu}(\mathbf{r})$ and $\rho_{2\nu}(\mathbf{r})$ associated with each site ν. If these functions represent the self-consistent density at geometry $\mathbf{P}^{(0)}$, a natural choice for the guessed density $\tilde{n}_\mathbf{P}(\mathbf{r})$ at a different geometry $\mathbf{P}$ is that $\rho_{1\nu}$ and $\rho_{2\nu}$ are carried along rigidly with the moving atoms while n_0 is unchanged.

Using the results of the previous sections, this choice leads to a simple force theorem. When the core states are treated separately, the Harris energy takes this form:

$$E_\mathrm{H} = \sum \epsilon_n^\mathrm{val} - \int n^\mathrm{val} V_\mathrm{eff} + U + E_\mathrm{xc} + \tilde{T}_\mathrm{core} \qquad (3.44)$$

where $\tilde{T}_\mathrm{core}$ equals $\sum \epsilon_i^\mathrm{core} - \int n^\mathrm{core} V_\mathrm{eff}$ and all eigenvalues are calculated in the effective potential V_eff, made from the input density $n^\mathrm{val} + n^\mathrm{core}$.

Integrals (including the implicit ones in U and E_{xc}) are all assembled out of contributions from the mesh density and from the spheres.

The aim is to derive the first-order change δE_{H} for the density change defined above. It reasonably straightforward that the last two terms give no contribution. Furthermore, all integrals over the spheres do not contribute. This is because the total-energy terms and the augmentation matrices for each atomic sphere are independent of the environment, depending only on the (here unchanged) $\rho_{1\nu}$ and $\rho_{2\nu}$. On the other hand, a lot of action happens because the electrostatic potential on the mesh changes by some amount $\delta\tilde{V}_0$ as the compensating gaussians g_ν move along with the atoms. In order to proceed, each compensating gaussian g_ν must be split into a valence part g_ν^{val} and a smooth representation of the core and nucleus g_ν^{cn}. Then the compensated mesh density is

$$\tilde{n}_0 = n_0 + \sum_\nu g_\nu = \Big[n_0 + \sum_\nu g_\nu^{\mathrm{val}}\Big] + \sum_\nu g_\nu^{\mathrm{cn}} \tag{3.45}$$

The term in brackets is the smooth representation of the valence density.

The contributions to δE_{H} from the first three terms of (3.44) are obtained as follows:

(1) By first-order pertubation theory, each eigenvalue $\epsilon_n^{\mathrm{val}}$ changes by

$$\delta\epsilon_n^{\mathrm{val}} = < C_n|\delta H - \epsilon_n^{\mathrm{val}}\delta S|C_n > \tag{3.46}$$

where H and S are the Hamiltonian and overlap matrices and C_n is the column eigenvector. By summing over the occupied states and taking a close look at how $\tilde{V}_0$ enters into the Hamiltonian, the outcome is

$$\delta \sum \epsilon_n^{\mathrm{val}} = \int \delta\tilde{V}_0 \Big[n_0 + \sum_\nu g_\nu^{\mathrm{val}}\Big] + \delta^{\mathrm{R}} \sum \epsilon_n^{\mathrm{val}} \,. \tag{3.47}$$

Here δ^{R} refers to the eigenvalue change when the potential $\tilde{V}_0$ and the augmentation matrices entering into H and S are kept frozen.

(2) The change in the second term is

$$-\,\delta \int n^{\mathrm{val}} V_{\mathrm{eff}} = -\int \Big[n_0 + \sum_\nu g_\nu^{\mathrm{val}}\Big]\delta\tilde{V}_0 - \int \tilde{V}_0\,\delta g_\nu^{\mathrm{val}} \,. \tag{3.48}$$

(3) Finally, the change in the third term is

$$\delta U = \int \tilde{V}_0 \Big[\delta g_\nu^{\mathrm{val}} + \delta g_\nu^{\mathrm{cn}}\Big] \,. \tag{3.49}$$

By summing these three contributions, the final force theorem comes out in the form

$$\delta E_{\mathrm{H}} = \int \tilde{V}_0 \, \delta g_\nu^{\mathrm{cn}} + \delta^{\mathrm{R}} \sum \epsilon_n^{\mathrm{val}} \; . \tag{3.50}$$

The first term describes the force of the smooth density on the gaussian lumps which represents the core and the nucleus at each site. This is in fact the same expression as a pseudopotential calculation would use.

The second term is a generalized Pulay term. It describes the eigenvalue shifts for a changing geometry but fixed smooth mesh potential and augmentation matrices. However, these augmentation matrices are used at shifted atomic positions. This force is not strictly the standard Pulay term [15], which is defined via the energy change when the potential (as a function in real space) is fixed. Rather, the term here is something which can be evaluated in a straightforward manner. It mainly involves the gradients of the various quantities (such as the expansion coefficients of the envelope functions) which are needed to assemble the Hamiltonian and overlap matrices.

3.3 Overview of the Implementation

At this stage, it is clear that we want to use smooth Hankel functions as envelopes and that they are to be augmented using the formalism presented above. This section summarizes the steps performed inside the program to implement this aim.

3.3.1 Density and Potential

At the start of each iteration, the input is the crystal charge density. As described, this is given by a smooth function $n_0(\mathbf{r})$ and local "true" and "smooth" $\rho_{1\nu}(\mathbf{r})$ and $\rho_{2\nu}(\mathbf{r})$ contributions associated with each site:

$$n(\mathbf{r}) = n_o(\mathbf{r}) + \sum_\nu \Big\{ \rho_{1\nu}(\mathbf{r}) - \rho_{2\nu}(\mathbf{r}) \Big\} \; . \tag{3.51}$$

In the program, the smooth density n_0 is represented using a regular mesh extending through the unit cell. Since several important operations within the loop over k points involve this mesh, the number of mesh points is a critical parameter. For each sphere, the two local contributions are represented on a radial mesh extending from the center of the sphere to the muffin-tin radius. This radial mesh must be fine enough to describe the oscillations of the all-electron wavefunction for the atom of interest. For this reason, the mesh is exponential with a closer spacing near the nucleus and the number of mesh points is selected according to the atomic number. The radial mesh is rather overkill for the smooth density $\rho_{2\nu}$, but for consistency all sphere contributions are represented in the same way.

To make the potential, the first step is to calculate the multipole moments of each sphere density, including the core and nucleus:

$$q_{\mu M} = \int_{S_\mu} \Big(\rho_{1\nu} - \rho_{2\nu}\Big)\, r^m Y_M(\hat{r})\, d\mathbf{r} + \frac{\delta_{M0}}{\sqrt{4\pi}}(Q_{\rm core} - Z) \ . \tag{3.52}$$

These are needed to make the compensated smooth density

$$\tilde{n}_0(\mathbf{r}) = n_0(\mathbf{r}) + \sum_{\mu M} q_{\mu M} G_{\mu M}(\mathbf{r}) = n_0(\mathbf{r}) + \sum_{\mu} g_\mu(\mathbf{r}) \ . \tag{3.53}$$

Here $G_{\mu M}$ is a gaussian of angular momentum M, centered on site μ and localized inside the muffin-tin sphere except for a negligible tail. It should be spread out enough so that it can be represented using the radial mesh inside the sphere, which should not be a problem. However, it does *not* have to representable on the rectangular mesh extending through the unit cell.

This last point is significant. For fast execution, it is important to keep the real-space mesh comparatively coarse. The starting point is that the smooth-Hankel envelope functions of the basis will be represented on this mesh. After that, things must be arranged in such a way that the mesh is not needed for *any* functions which are less smooth. Otherwise, a large part of the gain when using smooth Hankel functions disappears. For example, any gaussian which is localized *inside* an atomic sphere is of a different degree of smoothness than the smooth Hankels which extend smoothly *through* the sphere.

An important example is the electrostatic potential for the smooth compensated density:

$$\tilde{\phi}_0(\mathbf{r}) = \phi_0(\mathbf{r}) + \sum_{\mu} \phi\mu(\mathbf{r}) \tag{3.54}$$

where

$$\begin{aligned} \Delta\phi_0(\mathbf{r}) &= -8\pi\left[n_0(\mathbf{r}) - \bar{n}_0\right] & (3.55)\\ \Delta\phi_\mu(\mathbf{r}) &= -8\pi\left[g_\mu(\mathbf{r}) - \bar{g}_\mu\right] \ . & (3.56) \end{aligned}$$

In the last two equations, $\bar{n}_0$ and $\bar{g}_\mu$ are the average values over the unit cell. They must be subtracted because a periodic solution of Poisson's equation is only possible for a density which averages to zero. Since the full $\tilde{n}_0$ is neutral, the extra background charges must sum up to zero in the end, so that

$$\Delta\tilde{\phi}_0(\mathbf{r}) = -8\pi\tilde{n}_0(\mathbf{r}) \ . \tag{3.57}$$

The potential ϕ_0 of (3.54) is easily obtained by multiplying the **G**th Fourier coefficient of n_0 with $8\pi/G^2$. Each ϕ_μ is given analytically as a sum of smooth Hankels with energy equal to zero. The smooth electrostatic energy is

$$\begin{aligned} U_{\mathbf{smooth}} &= \tfrac{1}{2}\int \tilde{\phi}_0 \tilde{n}_0 \, d\mathbf{r} \qquad (3.58) \\ &= \tfrac{1}{2}\int \phi_0 n_0 \, d\mathbf{r} + \sum_\mu \int \phi_0 g_\mu \, d\mathbf{r} + \tfrac{1}{2}\sum_{\mu\nu} \int \phi_\nu g_\mu \, d\mathbf{r} \; . \end{aligned}$$

The first two integrals can be evaluated on the real-space mesh (or equivalently, in Fourier space) since each involves at least one smooth factor, but the last term is be done analytically.

The final smooth potential $\tilde{V}_0$ is obtained by simply adding the exchange-correlation potential of the smooth density:

$$\tilde{V}_0(\mathbf{r}) = \tilde{\phi}_0(\mathbf{r}) + \mu_{\mathrm{xc}}(n_0(\mathbf{r})) \qquad (3.59)$$

done together with the obvious exchange-correlation energy term

$$E^{\mathrm{xc}}_{\mathrm{smooth}} = \int n_0(\mathbf{r})\epsilon_{\mathrm{xc}}(n_0(\mathbf{r}))\, d\mathbf{r}. \qquad (3.60)$$

Purists might complain that $\tilde{V}_0$ is not the effective potential for any specific density, since the exchange-correlation part does not contain the compensating charges. However, this is irrelevant because everything inside the spheres will be substituted by the true functions when the sphere terms are added. The important point is to subtract out exactly those quantities which were added on for the real-space mesh.

To complete the construction of the potential and the energy integrals, the terms from the muffin-tin spheres are calculated and added on. The end result is a representation similar to that of the density:

$$V(\mathbf{r}) = \tilde{V}_0(\mathbf{r}) + \sum_\nu \Big\{ V_{1\nu}(\mathbf{r}) - \tilde{V}_{2\nu}(\mathbf{r}) \Big\} \qquad (3.61)$$

where $\tilde{V}_0$ is the smooth potential defined in the whole unit cell and $V_{1\nu}$ and $\tilde{V}_{2\nu}$ are the true and smooth sphere potentials. A tilde indicates that a potential includes a term from the compensating gaussians. As described in detail in Section 3.2.2, the sphere contributions are completely independent of the environment and depend only on the two density functions $\rho_{1\nu}$ and $\rho_{2\nu}$ in the relevant sphere. In practical terms, this means the boundary conditions on the surface of the sphere are set to zero when solving the Poisson equation for these two densities.

3.3.2 Augmentation Matrices

As described in Section 3.2.3, augmentation is done by means of small local matrices $\sigma_{kk'l}$, $\tau_{kk'l}$, and $\pi_{kk'LL'}$ defined for each site. These characterize the effect of the atom on the overlap, kinetic energy, and potential matrix elements. For example, the local matrix for the overlap is

$$\sigma_{kk'l} = \int_S \Big\{ \tilde{P}_{kL}\tilde{P}_{k'L} - P_{kL}P_{k'L} \Big\}\, d\mathbf{r} \qquad (3.62)$$

where the P_{kL} are functions used to expand the smooth envelopes around a site and $\tilde{P}_{kL}$ is a numerical solution of the Schrödinger equation which matches to value and slope on the sphere surface.

The general equations used to calculate the local matrices have been presented above. Two points still need clarification: how the numerical replacement for a smooth function is defined exactly, and which functions P_{kL} are used to expand the envelopes.

Phi and Phi-Dot

The aim is to construct a numerical solution $\tilde{f}_L(r)$ of the radial Schrödinger equation which matches the value and slope of a given function $f_L(r)$ with angular momentum l. Although the potential in the sphere generally has nonspherical terms, the numerical solutions are calculated for the spherical potential $\overline{V}(r)$:

$$\frac{1}{r}\frac{d^2}{dr^2} r\tilde{f}_L(r) = \Big[\frac{l(l+1)}{r^2} + \overline{V}(r) - \epsilon\Big]\tilde{f}_L(r)\ . \qquad (3.63)$$

This equation can be easily and accurately solved on the radial mesh, for example using the Numerov method. Note that the non-spherical potential terms are included when the local potential matrix $\pi_{kk'LL'}$ is calculated.

In principle, the matching could be done by adjusting the energy ϵ and the normalization of $\tilde{f}$. A better approach is to write $\tilde{f}$ as a linear combination of two independent functions and to adjust the two coefficients. As is usual in the LMTO method, these two functions are the radial solution at some fixed energy $\bar{\epsilon}_l$, normalized to one, and its energy derivative. These are invariably denoted by ϕ and $\dot{\phi}$, so that

$$\tilde{f}_L(r) = A_L\phi_l(r) + B_L\dot{\phi}_l(r)\ . \qquad (3.64)$$

Other choices are possible; for example, in the ASW method [16] the radial solutions at two different energies are used.

Up to now, the energies $\bar{\epsilon}_l$ in the radial Schrödinger equation are arbitrary, and indeed the eigenstates of the final matrix problem depend on them only weakly over a wide range. For optimal accuracy, the $\bar{\epsilon}_l$ should lie in the middle of the energy range of interest. This is done by placing them at the center of mass of the occupied part of the partial density of states (DOS) for each site and angular momentum. In practice, the DOS is not known until an iteration is completed, so the center of mass is taken as $\bar{\epsilon}_l$ for the following iteration.

For a given potential, a normalized solution of the radial Schrödinger equation can be characterized uniquely either by the energy or, equivalently, by the number of nodes and the boundary conditions at R_{mt}. The boundary conditions are given by the logarithmic derivative

$$D_l = \frac{R_{\mathrm{mt}}\phi_l'(R_{\mathrm{mt}})}{\phi_l(R_{\mathrm{mt}})} \; . \tag{3.65}$$

For various reasons, it is convenient to specify the radial solutions $\phi_l(r)$ in this way, rather than by prescribing the energies $\bar{\epsilon}_l$ directly. For example, energies can shift up or down significantly during the self-consistency iterations, while the logarithmic derivative is more stable. Also, it is easier to guess a suitable starting value for the boundary conditions than for the energy. The number of nodes gives the principal quantum number, so this and the logarithmic derivative are combined into one parameter by the definition

$$P_l = (\mathrm{P.Q.N.}) + \left\{ \tfrac{1}{2} - \arctan(D_l)/\pi \right\} \; . \tag{3.66}$$

The integer part of P_l is the principal quantum number and the fractional part translates into the logarithmic derivative. As the energy $\bar{\epsilon}_l$ increases, P_l also increases monotonously, the logarithmic derivative becomes more negative, and the associated wavefunction approaches a state which has a node at the muffin-tin radius. This state is formally reached when P_l is an integer value, crossing the border to the next-higher principal quantum number.

By choosing $\bar{\epsilon}_i$ in the center of the partial DOS, good accuracy is ensured in most cases. Sometimes, however, it leads to drastically wrong results. A characteristic situation is when it is not obvious which principal quantum number to use for the valence states in some L component. For example, the $3p$ states might be some distance below the valence band while the $4p$ states lie above the Fermi level. It is then often sensible to include the high-lying states (here: $4p$) as valence degrees of freedom. The problem is that, by construction, the occupied part of the DOS lies below the $4p$ states. When $\bar{\epsilon}_p$ is positioned there, the corresponding ϕ_p and $\dot{\phi}_p$ can sometimes combine to a function which belongs to the lower principal quantum number. This manifests itself in spurious low-lying eigenvalues, also called “ghost states,” and by a value of P_l only slightly above the selected principal quantum number. Luckily, the solution is simple: just keep the corresponding P_l frozen at a reasonably large value.

Expansion of Smooth Envelopes

In Section 3.1.5 it was described how a smooth Hankel envelope function can be expanded as a sum of the polynomials $P_{kL}(\mathbf{r})$ around a site. The coefficients of the expansion are given by the integrals of the smooth Hankel function times the higher-order gaussians G_{kL} centered at the site. These integrals can be done analytically [10].

In principle, this leads to an elegant approach in which there is no need to distinguish between the site where the envelope is centered (the "head") and all the other sites (the "tails" of the function). The expansion as a sum of P_{kL}'s is equally valid for both, as is the expression for the expansion coefficient. However, it turns out that this is numerically wasteful. Depending on the smoothing radius of the function, it takes considerably more terms in the sum over k to expand the head than the tail at another site. Thus, a better solution is to use the explicitly known form of the head function. Formally, this means that the set used to expand the envelopes at a certain site consists of two types of functions, namely the polynomials P_{kL} together with the heads centered there:

$$\{P_{kL}\} \to \{P_{kL}, H_{jL}\} \ . \tag{3.67}$$

The formalism for the augmentation proceeds exactly as before, except that three different local matrices must be made. For the overlap, these are the matrices $\sigma^{PP}_{kk'l}$, σ^{PH}_{kjl}, and $\sigma^{HH}_{jj'l}$. The first involves a pair of polynomials P_{kL}, the second the product of a polynomial with a head function, and the third the product of two Hankel heads.

3.3.3 Calculation of Eigenstates

In a completely standard way, the eigenvalue problem is solved by the Rayleigh-Ritz variational method. Each eigenfunction is expressed as linear combination of the basis functions $\tilde{H}_i(\mathbf{r})$:

$$\psi_n(\mathbf{r}) = \sum_i B_{in} \tilde{H}_i(\mathbf{r}) \ . \tag{3.68}$$

In the present case, the basis consists of smooth Hankel functions (denoted by H_i), centered at the various sites of the system and augmented inside all atomic spheres (expressed by the tilde over H_i). Each basis function is specified by the following information: the site ν where it is centered, its angular momentum L, its decay at large radii expressed as an energy ϵ, and how strongly it bends over near the central site, expressed by the smoothing radius R_{sm}. This whole set $(\nu, L, \epsilon, R_{\text{sm}})$ has been contracted to one majestic index i. The coefficients of the wavefunction are obtained by solving the generalized matrix eigenvalue problem

$$HB_n = E_n S B_n \tag{3.69}$$

for the column eigenvector B_n of the n-th wavefunction and the corresponding eigenvalue E_n. The Hamiltonian $H = T + V$ and overlap matrices are defined as

$$\begin{aligned} H_{ij} &= \int \tilde{H}_i^*(\mathbf{r})[-\Delta + V_{\text{eff}}(\mathbf{r})]\tilde{H}_j(\mathbf{r})\, d\mathbf{r} && (3.70) \\ S_{ij} &= \int \tilde{H}_i^*(\mathbf{r})\tilde{H}_j(\mathbf{r})\, d\mathbf{r}\ . && (3.71) \end{aligned}$$

These are described below.

3.3.4 Secular Matrices

Section 3.2.3 explained how to augment an arbitrary smooth envelope function. Written out in full splendor for $H_i(\mathbf{r})$, the result is

$$\tilde{H}_i(\mathbf{r}) = H_i(\mathbf{r}) + \sum_\mu \sum_{kL} C_{i\mu kL}\Big\{\tilde{P}_{\mu kL}(\mathbf{r}) - P_{\mu kL}(\mathbf{r})\Big\} \tag{3.72}$$

where $C_{i\mu kL}$ is the kL-th coefficient of the local expansion of H_i around site μ. That is: H_i is first approximated as a finite sum of the functions P_{kL}, consisting of the polynomials of Section 3.1.5 and the on-site heads. To obtain $\tilde{H}_i$, each P_{kL} in the expansion is replaced by an appropriate numerical solution of the radial Schrödinger equation, denoted by $\tilde{P}_{kL}$.

The matrix elements for the overlap, kinetic energy, and potential are all calculated in the same general form. They involve the vectors $C_{i\mu}$ of the expansion coefficients and the local augmentation matrices σ_μ, τ_μ, and π_μ for each site:

$$\begin{aligned} S_{ij} &= \int H_i^* H_j\, d\mathbf{r} + \sum_\mu C_{i\mu}^\dagger \sigma_\mu C_{j\mu} && (3.73) \\ T_{ij} &= \int H_i^*(-\Delta) H_j\, d\mathbf{r} + \sum_\mu C_{i\mu}^\dagger \tau_\mu C_{j\mu} && (3.74) \\ V_{ij} &= \int H_i^* \tilde{V}_0 H_j\, d\mathbf{r} + \sum_\mu C_{i\mu}^\dagger \pi_\mu C_{j\mu} && (3.75) \end{aligned}$$

where H_i and H_j are the unaugmented envelopes, *i.e.*, smooth Hankel functions centered on the various atoms.

The general procedure thus is as follows. We first calculate the corresponding matrix element for the smooth unaugmented functions, which is the first term in each equation. Then, local information at the various sites is projected out in the form of the vectors $C_{i\mu}$. These vectors are combined with the local augmentation matrices to make the second term.

A pleasant feature is that the overlap and kinetic energy integrals for the unaugmented functions can be done analytically [10]. These terms reduce to sums of smooth Hankel functions evaluated for the connecting vector between the two sites, which are then calculated using Ewald summation.

The augmentation contributions are also straightforward. When expanding a given function as a sum of the polynomials P_{kL} near a site μ, each coefficient is the integral over the function times the gaussian G_{kL} centered at the site. This is the step which projects out the local information about the envelopes. Thus we need integrals of the form

$$\int G^*_{kL}(\mathbf{r}-\mathbf{R}_\mu)H_{L'}(\mathbf{r}-\mathbf{R}_\nu)\,d\mathbf{r} \tag{3.76}$$

Again, these reduce to smooth Hankel functions evaluated for the connecting vectors $\mathbf{R}_\mu - \mathbf{R}_\nu$. They are calculated by Ewald summation in the program.

This leaves the matrix elements of the smooth potential for the unaugmented functions, which need a bit of thought:

$$V^{\text{smooth}}_{ij} = \int H^*_i \tilde{V}_0 H_j \, d\mathbf{r}\ . \tag{3.77}$$

The plan is to calculate these by numerical integration. At the risk of being repetitive, the basic idea was that this is permissible because the H_i are relatively smooth functions. To tabulate the envelope function on the real-space mesh, we set up the Fourier transform in reciprocal space and do a fast Fourier transform to real space. The Fourier transform of a smooth Hankel function with energy ϵ, smoothing radius R_{sm}, and center $\mathbf{R}_\nu$ can be written down directly:

$$\hat{H}_L(\mathbf{q}) = -\frac{4\pi}{\epsilon - q^2} e^{\gamma(\epsilon - q^2)} \mathcal{Y}_L(-i\mathbf{q}) e^{-i\mathbf{q}\cdot\mathbf{R}_\nu} \tag{3.78}$$

where $\gamma = 1/4R^2_{\text{sm}}$.

The simplest procedure would be to set up the Fourier transforms of H_i and H_j separately, transform both to the real-space mesh, and then sum $H^*_i\tilde{V}_0H_j$ over the mesh points. This can be done, but care must be taken in the proper definition of the mesh potential $\tilde{V}_0$ used here. The difficulty is that the potentials of the "pseudonucleii," *i.e.*. the localized gaussians which represent the nucleii and the cores, are too strongly localized to be represented on the real space mesh. Therefore, $\tilde{V}_0$ is made by setting up its Fourier transform including the pseudonucleii, truncating this to the low cutoff suitable for the smooth Hankel functions, and transforming to the real space mesh. For each mesh point, this potential is not the same as a point-wise evaluation of $\tilde{V}_0$ according to its definition (3.59). Instead, the mesh potential has the useful property that it returns the correct integral when used in a scalar product with any smooth function.

The remaining problem is that the described procedure involves too many fast Fourier transform (FFT) steps. A much more efficient way to arrange things is as follows. We loop over the functions H_i in the basis set. For each one, the Fourier transform is set up, a FFT is applied to tabulate the function on the real-space mesh, the product $H_i(\mathbf{r})\tilde{V}_0(\mathbf{r})$ is made, and this is transformed back to reciprocal space. Then, a second loop runs over the basis functions H_j, making the scalar product between the Fourier transforms of $H_i\tilde{V}_0$ and H_j to produce the integral.

Chapter 4

Operating the Program

This chapter explains how to use the program to perform a calculation. After an overview, the installation starting from the source code is described. Then, the syntax and the significance of the various input parameters are considered in detail. Finally, the steps needed to set up and run a calculation are explained.

4.1 Overview

People who just want to get on with it have presumably skipped directly to this chapter. For this reason, the underlying ideas are summarized once more, followed by an overview of the main programs and I/O files.

4.1.1 Program Capabilities and Function

This program calculates the total energy of a crystalline system, given the positions of the nucleii in the unit cell. The interaction between electrons is treated by the local-density approximation within density-functional theory. This is an all-electron method, which means that the core states are calculated explicitly and that the valence wavefunctions exhibit the oscillations required for orthogonality to the core. All atoms of the periodic table can be treated in a similar manner and with comparable efficiency. The program is "full potential," *i.e.*, there is no restriction on the shape of the potential and charge density.

The present program grew out of a previous FP-LMTO method [17] which used a fitting procedure in the interstitial region. The older method is reasonably accurate and fast, but there are two severe limitations: "empty spheres" must be added for structures which are not close-packed, and the forces on the atoms are not calculated. In the development of the current program, elimination of these two problems was the main priority. A further aim was the simplification of the augmentation procedure, inspired by the simplicity of the pseudopotential approach.

The main bottleneck in a full-potential method using some kind of local orbitals is to evaluate the matrix elements of the interstitial potential. There are some advantages to a straightforward numerical integration over the unit cell, followed by a subtraction of the sphere contributions, whereby the integrand is extended smoothly through the spheres. Indeed, good FP-LMTO programs which work in this way already exist. Unfortunately, the problem with a numerical integration is that the effort is usually very large, soon swamping out the other steps in the calculation.

The solution implemented here is to use a new type of envelope function, namely atom-centered Hankel functions which have had their singularity removed. The functions bend over as they approach the central muffin-tin sphere. This behavior is similar to that of the the true wavefunctions, allowing a smaller basis set. Furthermore, these "smoothed Hankel functions" can be integrated using a coarser mesh. Together, these two properties lead to a substantial increase in the program efficiency. If the basis is tuned carefully, the execution time becomes comparable to the older program. A user should understand the connection between the parameters which define the basis set and the required real-space mesh in order to exploit this saving.

It turns out that the forces drop out more or less naturally from the new formulation of the augmentation step. The basic idea is to write quantities consistently as a smooth function extending through the spheres, to which local terms are added inside each sphere. The resulting formulation has some similarity to a unique and transferable pseudopotential and permits considerably lower angular-momentum cutoffs than previously.

4.1.2 Main Programs and I/O Files

The central program `nfp` contains the main self-consistency loop. Some other programs are also needed, for example to prepare the free atoms and to generate k points for the Brillouin-zone integrations. In addition, auxillary programs help to set up a calculation and to make plots.

The following main programs are included:

`nfa`	program to calculate free atoms
`nqp`	program to set up a list of k points
`nfp`	program for the main self-consistency loop
`nchk`	program to check the input parameters
`nsymf`	auxillary program to determine symmetry operations

`nsup`	auxillary program to construct supercells
`nband`	program for a band-structure plot
`nden`	program for a charge-density plot

The input and output files are identified by their extensions. For example, a job might have the input file `si.mas` and would write the files `si.log`, `si.qpt` etc. The various extensions have the following meaning:

`mas`	main input file.
`atm`	free-atom data.
`qpt`	list of k points
`log`	log file
`rst`	restart file
`sum`	one-line summaries
`bnd`	data for band-structure plot
`chd`	data for charge-density plot
`dos`	density of states
`bs`	data for ball-and-sticks structure plot
`tmp`	temporary file for extra data

To perform a calculation, the first step is to set up a valid input file `<id>.mas`. The free atoms are calculated to supply a first guess at the crystal charge density when overlapped. The free-atom data is written to `<id>.atm`. A separate program `nqp` supplies the k points and weights for the Brillouin-zone integration, written to `<id>.qpt`. When the parameters which determine the k-space mesh are changed in the input file, program `nqp` must be run again, or the change will not come into effect.

The main selfconsistency loop is done in program `nfp`, as shown in the flow chart in Fig 4.1. This program reads `<id>.atm` and `<id>.qpt`. It also reads and/or writes the restart file `<id>.rst` as determined by switches in the main input file. This makes it possible to do further iterations in a later job and to save the data from a self-consistent calculation. In addition, most of the programs write important data to the log file `<id>.log` in compact form. New output is attached to the end of an existing log file, so that the this file grows continually. In a similar way, a one-line summary of each calculation is appended to the summary file `<id>.sum`. A user might want to change the lines which define this information near the end of file `nfp.m.f`.

Finally, for making plots, `nfp` writes the DOS to `<id>.dos` if requested, the check program makes the ball-and-sticks input file `<id>.bs`, and separate programs generate `<id>.bnd` and `<id>.chd`.

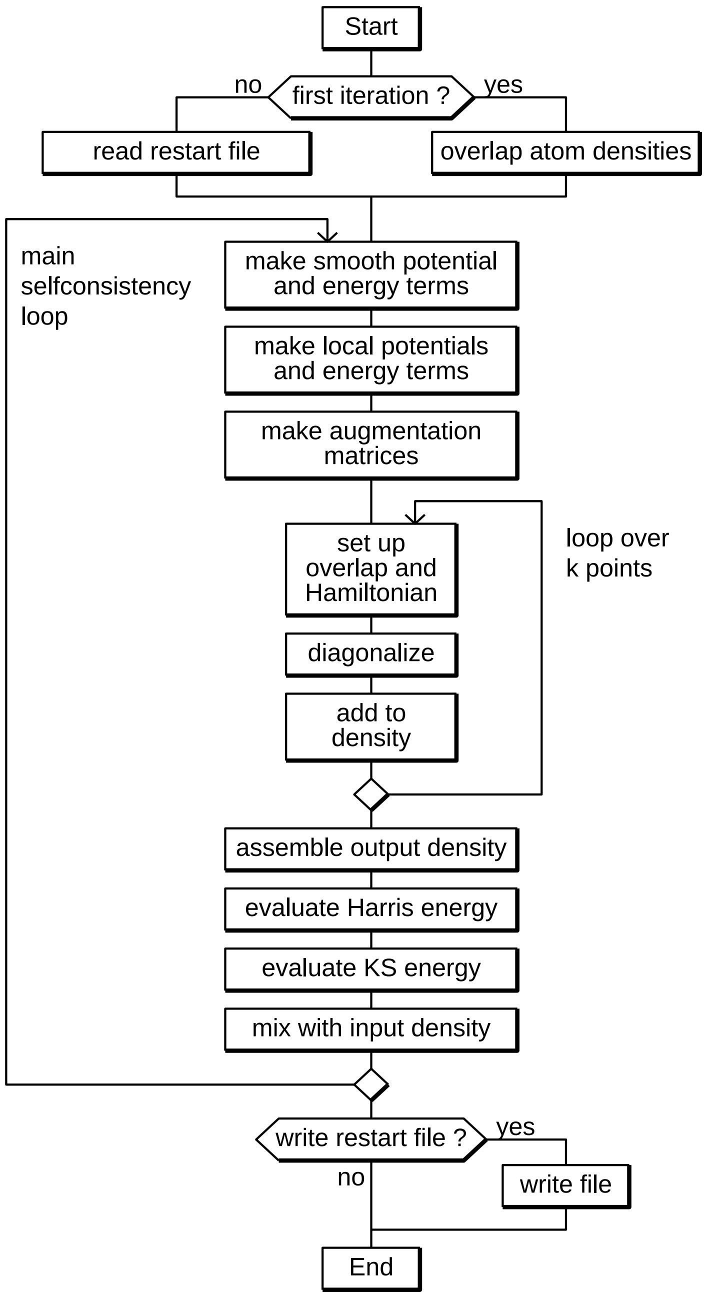

Figure 4.1: Flow chart of the program containing the main selfconsistency loop, `nfp`. There is nothing unusual here; this program flow is positively old-fashioned in comparison to "wheels within wheels" plane-wave programs. Note that the block "evaluate KS energy" in fact repeats the operations "make smooth potential..." and "make local potentials..." for the output density.

4.2 Installation

In this package, files containing a main program are identified by the extension `.m.f`, for example `nfp.m.f` and `nfa.m.f`. Subroutines are identified by the extension `.f`, for example `wkinit.f`. A subroutine is usually in a file by itself, but the file may contain additional "private" routines called only by this routine. The program is written mainly in standard FORTRAN 77. All floating-point arithmetic is done in double precision using `REAL*8` and `COMPLEX*16` variables. The intellectual challenge of devising meaningful six-letter names for variables and subroutines proved too great; a compiler should be able to distinguish names of up to about 12 character in length. The `IMPLICIT NONE` statement is used occasionally.

4.2.1 Compilation

There is no standard way to organize source code and executable programs into directories, but one convenient system is as follows:

1. Put all source code files into one directory, typically `nfp/f`.
2. Compile all the subroutines and put their object codes into a library `lib.a` in the same directory.
3. Compile all main programs and link them with `lib.a` (and possibly other libraries) to make executables with the extension `.x`.

After these steps, the directory will contain the source code, the library `lib.a`, and the executables `nfa.x`, `nqp.x` etc. There is a shell script `MAKE` to help in this process. The comments near the top of `MAKE` explain how to use it.

Before installation, these points must be considered:

- There are a few routines which are system-dependent, for example those concerned with file handling, retrieving information about the operating system, and timing. These are isolated in one file named `syscalls.f`. When installing on a new system, these routines must presumably be modified.
- Each main program allocates and releases memory from one long workspace array, dimensioned as a real array of length `nsize` in the main program. Arrays of type `REAL*8`, `COMPLEX*16`, and `INTEGER` as well as `REAL*4` are allocated as chunks of suitable length. A few overzealous compilers (Burroughs?) forbid passing a subroutine a real

array and then using it as another type, say `REAL*8`. In such a case installation will not be easy.

Parameter `nsize` must in any case be large enough for a given calculation. Depending on the operating system, it may be disadvantageous to choose it much larger than needed. To help choose a value, the actually used workspace is printed in the listing and the log file. If a trace profile is requested, the workspace at the entry and exit of important subroutines is shown.

On some computers (Cray?) single-precision corresponds to double-precision in normal usage. For these, a compiler flag can usually be set to avoid compilation in double precision. If this is used, the allocation routines in `wkinit` should be changed accordingly. Otherwise, all floating-point workspace arrays will be dimensioned twice as large as needed.

- Two other dimensioning parameters which might have to be changed in the main programs are `nbx`, the maximal number of atoms, and `nsx`, the maximal number of species. These can be chosen generously since they do not allocate very much memory.

- For the program speed, the fast Fourier transform (FFT) routine is significant. The FFT is packaged in a way which makes it easy to slot in a different routine. If an optimized routine for a three-dimensional double-precision complex FFT is available, it is probably a good idea to change `fft_c3.f`. This file contains `fft_c30`, which returns the dimensions for an array to be transformed, and `fft_c3`, which calls the actual FFT routine.

4.2.2 Running the Programs

The programs take the identifier for the input/output files from environment variable `LMJOB`. Generally a calculation will be done in some working directory, and *not* in the directory containing the sources and executables. The direct procedure to run, say, programs `nfa.x` and `nqp.x` for an input file `si.mas` in some directory would be by typing

```
setenv LMJOB si
~/nfp/f/nfa.x
~/nfp/f/nqp.x
```

where `~/nfp/f` refers to the directory with the executable programs.

To do this slightly more elegantly, a shell script `n` is available. It checks the existence of the input file and the executables, takes a local executable if there is one in the current directory, and permits some useful options. This script should be placed in a directory such as `~/bin` which is in the path. Near the top of the script, the line defining the directory containing the executables should be set properly. Typing `n` without additional arguments should supply details about the usage.

It would be nice to combine all functions needed to run the programs in one command. Shell script `n` can be used to show the job identifier `LMJOB`, but due to the quirks of UNIX it cannot be used to replace the somewhat unwieldy line which sets `LMJOB`. However, a convenient alias can be defined in the `.cshrc` file by the following line:

```
alias njob 'if (X\!* != X) setenv LMJOB \!* ; echo $LMJOB'
```

This defines `njob` to print out the current setting of `LMJOB` if entered without an argument, and to set `LMJOB` to a given identifier if used with an argument. Using these shortcuts, the commands above can be replaced by

```
njob si
n fa
n qp
```

or by the single line

```
n -id si fa qp
```

4.2.3 Some Programming Details

For those who might want to understand the code, here are some additional details.

Workspace

The workspace array, introduced above, is organized strictly as a stack. This means that a newly-allocated array is placed behind all those allocated previously, and that the release of an array also releases all those defined later. A routine which defines a temporary array returns the index of the first element of the array. These indices play a role similar to pointers in C. For consistency, they are all denoted by names starting with the letter `o`, defined as an implicit integer. Internally, the routines which handle the workspace maintain the index of the first element not yet allocated, called `ofree`. To define a new array, the current value of `ofree` is returned as the pointer to the array and `ofree` is advanced by the number of array elements. To release an array, `ofree` is simply set equal to the pointer to that array.

Somewhat inconveniently, the elements of a temporary array cannot be manipulated in the subroutine which defines it, but only in other routines called by it. This explains some of the "private" routines, called by only a single higher-level routine. Schematically, a routine which allocates a temporary array is along these lines:

```
subroutine sub1(a,b,c)
implicit real*8 (a-h,p-z), integer (o)
real w(1)
common w /w/
  ...
call defrr(oamat, 100*100)
call sub2(100,100,w(oamat))
call rlse(oamat)
  ...
end
```

The `real` and `common` statements access the workspace common block. The `defrr` statement allocates a new array of 100 times 100 doubleprecision elements, returning the pointer `oamat`. The call to the lower-level subroutine `sub2` passes through the first array element, `w(oamat)`. When the array is not needed any more, the call to `rlse` sets the "free" pointer back to `oamat`. Inside `sub2`, array `amat` is treated in the standard way:

```
subroutine sub2(m,n,amat)
implicit real*8 (a-h,p-z), integer (o)
dimension amat(m,n)
  ...
end
```

As an additional check, an extra element of the workspace `w(nsize)` is used at the start of each allocated array to point to the next array along the stack. At any time, these links can be run through as a sanity check by calling `wkchk`. The links are checked automatically whenever a new array is allocated. If a link has been destroyed, it is clear that an array has been overwritten and the program will stop.

Structs

Programming in FORTRAN is made difficult because information cannot be bundled together in the form of "structs" as in C. For example, it is conceptually obvious which parameters are needed to define the properties of an atomic species. These include the nuclear charge, the mass, the muffin-tin radius, the basis set, etc. In C, these would be addressed as `spec.z`, `spec.mass` and so on. In standard FORTRAN, each parameter is a separate variable. This unfortunately leads to (i) unpleasantly long argument lists for subroutines and (ii) the need to change whole chains of subroutine calls whenever a new parameter is added.

As a workaround in this program, parameters which belong together conceptually are stored as elements of a double-precision vector. Such a vector plays the role of a struct and is given a name such as `s_spec`. Each parameter is assigned a well-defined position in the vector. In the interest of program transparency, it is definitely a bad idea to refer to parameters directly, for example as `s_spec(12)`. Instead, special "pack" and "unpack" routines are called to store or retrieve parameters. For instance, routine `u_pack_a` extracts the parameters associated with augmentation from struct `s_spec`.

The following structs are used in the programs:

`s_spec`	data to define the atomic species
`s_atom`	data for the separate atoms
`s_ctrl`	data for overall program control
`s_lat`	data related to the crystal lattice
`s_dyn`	data about the atom dynamics
`s_etot`	various contributions to the total energy
`s_strn`	strings

When using the program, it is obviously irrelevant how the data is stored internally. The real advantage of using structs becomes clear when new features are being added. For example, assume we want to calculate the electric field gradients at the nucleii. For each atom, the 3×3 matrix of second derivatives of the electrostatic potential is required. This is the sum of several terms, to be evaluated in completely different subroutines. Without using the structs, a new array such as `fg(3,3,nbas)` would be defined in the main program. It would then be passed down different branches of the call hierarchy, making it necessary to change all subroutines along the way. Of course, we could also just introduce a new common block to pass the array through "invisibly." This, however, would be a major step down a slippery slope which ends with unmanageable program code.

Using structs, we simply assign nine elements of the atom struct for each site to the field gradient and write two new pack and unpack routines to access them. After that, the field gradient matrix can be inspected or changed in any subroutine in a clean way. A second closely related feature is that new parameters can easily be added to the input file without the need to change subroutine calls.

4.3 The Input File

This section describes the syntax of the input file and and the significance of the various input parameters systematically.

4.3.1 Units

This program uses atomic Rydberg units. Distances are measured in Bohr radii and energies in Rydberg, given as

$$\begin{array}{rcccl} a_0 & = & \hbar^2/me^2 & = & 0.529177\ \text{Å} \\ \text{Ry} & = & e^2/2a_0 & = & 13.6058\ \text{eV} \end{array} \tag{4.1}$$

The point of these units is that the Schrödinger equation has an especially simple form, namely

$$\left\{-\Delta - \frac{2Z}{r}\right\}\Psi(\mathbf{r}) = E\Psi(\mathbf{r}) \tag{4.2}$$

for the Coulomb potential of a nucleus with atomic number Z. For a given electron density $\rho(\mathbf{r})$, the repulsive electrostatic potential felt by an electron is

$$V_{\text{es}}(\mathbf{r}) = 2\int \frac{\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\, d\mathbf{r}' \tag{4.3}$$

which satisfies the Poisson equation

$$\Delta V_{\text{es}}(\mathbf{r}) = -8\pi\rho(\mathbf{r})\ . \tag{4.4}$$

4.3.2 Syntax of the Input File

There is one main input file to each job, historically called the master file and identified by the extension “mas”. The syntax is free-format: lines do not have to be in any special sequence and spaces can be added as desired. Within the file, variables can be defined and used in mathematical expressions. It is also possible to retrieve environment variables to control a calculation from the operating system.

Basic Syntax

Here is a minimal (but still valid) input file which demonstrates the basic points of the syntax.

```
%  Si sample input file

id        < Si >
control   {nit}  1  {add} 0  {beta} 0.6   {eks} 1
output    {verb} 30   {logstep} 1   {trace} 1  {profile} 1
switches  {rel} 1  {spinp} 0  {friz} 0  {xc} 1  {grad} 0
ftmesh    12 12 12
symgroup  < r3d  i r4z:t(.25,0,.25) >
kmesh     {job} 1  {nkxyz}  4 4 4
lattice   {a} 10.20   {vecs} 0 .5 .5    .5 .0 .5   .5 .5 .0
pos  <si>   -.125 -.125 -.125
pos  <si>    .125  .125  .125

% define species
spec  <si>
  gen   {z} 14.0  {mass} 1   {rmt} 2.20
  chd   {lmxl} 2  {rg} 0.55  {rsmv,kv}  0.95  15
  aug   {rsma} 0.90  {kmax} 4  {lmax} 3  {pnu} 3.5 3.5 3.3 3.15
  bas   'h' {l} 0 2 {rsm} 1.4  {e} -0.5
  bas   'h' {l} 0 1 {rsm} 2.2  {e} -1.4
  fa    {rsmfa} 1.25   {Qat}  2.0  2.00
endspec
```

The basic rules for the input syntax are as follows:

- Ignorable input (*i.e.*, comments) are empty lines, all characters from `%` to the line end, and all characters between braces `{...}`.
- The first word of each line is a keyword, defining the type of the line. Lines with an unrecognized keyword are silently ignored. The keyword can start in any column. Within a line, words are separated by an arbitrary number of spaces.
- Within each line of a specific type, the parameters are expected to be in a well-defined order. For some line types, a number of the parameters at the end of the line are optional.
- Lines can be in any sequence, but depending on its type, a line may appear only inside a species block or outside it.
- A certain subset of lines is required to run a job. Other lines can be included only as needed when the program defaults are to be changed. For example, a line with the keyword `ewald` will change the default parameters for the Ewald summation.

- Strings are coded using either angle brackets `<...>` or single quotes `'...'`. For example, the line with the keyword `id` defines an identifier string for the job.
- A backslash \ at the end of a line means that the next line is a continuation of this one.

The syntax is free format, in the sense that lines can be arranged in any order and that spaces do not matter. However, the sequence of parameters within each line is significant. All embedded strings `{...}` are for clarity only and do not influence the interpretation of the parameters in any way. Such strings can be changed to any desired text or left away. For example, the following lines are all equivalent:

```
control  {nit} 1  {add} 0  {beta} 0.6   {eks} 1
control  {iterations} 1  {more} 0  {mix} 0.6  {KS switch} 1
control  1  0  0.6  1
```

but the next line will *not* function in the expected way:

```
control  {beta} 0.6  {nit} 1  {add} 0  {eks} 1    % Wrong !
```

Extended Syntax

The described features are adequate to operate the program. For convenience, some additional functions are available. Variables can be defined and used, mathematical expressions can be evaluated, and environment variables can be retrieved from the operating system. An example follows:

```
%  extended Si input file

getenv ALAT
set dist=ALAT*sqrt(3)/4

id        < Si, alat="ALAT" >
control   {nit}  1  {add} 0  {beta} 0.6   {eks} 1
output    {verb} 30   {logstep} 1   {trace} 1  {profile} 1
switches  {rel} 1  {spinp} 0  {friz} 0  {xc} 1  {grad} 0
ftmesh    12 12 12
symgroup  < r3d  i r4z:t(.25,0,.25) >
kmesh     {job} 1  {nkxyz}  4 4 4
lattice   {a} ALAT   {vecs} 0 .5 .5    .5 .0 .5   .5 .5 .0
pos  <si>   -.125 -.125 -.125
pos  <si>    .125  .125  .125
```

```
% define species
spec  <si>
  gen   {z} 14.0 {mass} 1  {rmt} 0.98*dist/2
  chd   {lmxl} 2  {rg} 0.55  {rsmv,kv}  0.95  15
  aug   {rsma} 0.90  {kmax} 4  {lmax} 3  {pnu} 3.5 3.5 3.3 3.15
  bas   'h' {l} 0 2 {rsm} 1.4  {e} -0.5
  bas   'h' {l} 0 1 {rsm} 2.2  {e} -1.4
  fa    {rsmfa} 1.25   {Q} 2.0  2.00
endspec
```

This example retrieves the lattice constant from the environment variable `ALAT`. It then sets variable `dist` to the nearest-neighbor distance. Next, variable `ALAT` is used to set the lattice constant in the `lattice` line, and variable `dist` is used to calculate a nearly-touching muffin-tin radius in the `gen` line. Furthermore, the numerical value of `ALAT` is placed into the identifier string to make this line more informative, for example in the job log file. The example is a little unfortunate because the recommended procedure is actually to keep the muffin-tin radii fixed, but it serves to demonstrate the syntax when using variables.

The main reason to control the program execution using environment variables is to perform a series of related jobs using a shell script. A typical application would be to do calculations for a sequence of different lattice constants. This type of job control is explained in Section 4.5.2.

The exact syntax rules for the extended features are as follows:

- Mathematical expressions must be written in one string without embedded spaces. They can contain numbers, variable names, the binary operations `+ - * / ^`, and the following functions: `exp`, `log`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `sqrt`, `cbrt`, and `fix`*n* where *n* is an integer. An expression can be simply a number or the name of variable.

- Each variable and expression is of type floating point or integer. The type of an expression derived from the types of the constants and variables which enter. This is float if any the operands is of type float or if a function is used, otherwise it is integer. As usual, integer arithmetic is done without roundoff errors and with truncation of any fractional part (*e.g.*, 3/2 gives 1).

- An expression can take the place of a parameter in an input line. Parameters which expect an integer value must be substituted by an integer expression.

- The value of an expression can be embedded in a string by placing it in double quotes within the string.
- Variable names can be upper or lower case, must start with a letter, and can contain letters, digits, and the underscore character.
- To assign a variable, the file should contain the keyword `set` followed by one or more comma-separated assignements, each of the form name=expression. The type of the variable (int or float) is set equal to that of the expression. To force an integer assignment, `set` should be replaced by the keyword `int`. Except in rare cases, `set` will do the job.
- Environment variables are retrieved using the keyword `getenv` followed by one or more variable names, separated by commas. After being initialized in this way, they are treated like any other variables. Since environment variables are written in upper-case letters by UNIX convention, they will probably have capitalized names inside the input file also.
- If a requested environment variable is undefined or blank, the outcome of the `getenv` statement depends on whether the variable has already been assigned a value inside the input file. If this is the case, the program continues using this value. If not, the program stops with an error message. In this way a prior assignment can be used to set a default.
- A block starting with the line `job` and ending with the line `endjob` is ignored when the input file is parsed. The purpose is to include lines for the job control in the input file, which are extracted and executed by a suitable shell script (see Section 4.5.2).
- To help find errors during input file parsing, variable `trace` should be set. Depending how high its value is (up to about 3), more or less output is produced. Tracing can be switched off again by adding the line `unset trace`.

4.3.3 Input Parameters

The following table describes the parameters of the input file, grouped by the type of line in which they appear.

Data lines are of two types. Standard lines appear in the main body of the file and set general parameters for the job. Species lines appear inside a `spec` ... `endspec` block and are used to define an atomic species. Some lines are optional and can be left away completely. They are only needed to change the built-in defaults or to supply data for an auxillary program (*e.g.*, the symmetry lines for a band structure plot).

For a minimal job, the following lines are required in the main part of the input file:

```
id          output      control     switches    lattice
symgroup    ftmesh      kmesh       pos
```

and the following lines are needed to define at least one atomic species:

```
spec        gen         chd         aug
bas         fa          endspec
```

A few lines have optional parameters at the end of the line which can be left away. These parameters are indicated by parantheses in the table below.

As a reminder, the strings in braces `{...}` are arbitrary comments, used here to identify the parameters. Within each line, parameters are identified purely by their sequence.

Standard Lines

`id` `< Si cubic >`

Set the identification string.

`<..>`	identification string. Used in the listing, the log, and the restart file. Expressions and input file variables can be included by putting them in double quotes.

`output` `{verb} 30  {logstep} 1  {trace} 1  {profile} 1`

Control output to the listing and log.

`verb`	verbosity (between 0 and 100). Controls the amount of output to the listing. Standard value is about 30.
`logstep`	interval between iterations for which data is written to the log file.
`trace`	trace depth. Entry and exit into important routines is traced together with the cpu time to the specified depth.
`profile`	trace profile depth. Same function as `trace`, but the data is collected in a summary at the end of the listing.

`control` `{nit} 1 {add} 0 {beta} 0.6 {eks} 1`

Overall program control.

`nit`	upper iteration limit. Iterations are counted from one when starting from overlapped free atoms, and from the last completed iteration when starting from the restart file. Ignored if `add` is nonzero.
`add`	if nonzero, the number of additional iterations to do, counting from the last completed iteration.
`beta`	mixing parameter for the charge density ($0 < \beta < 1$). When $\beta = 1$, the unadulterated output density is used (which will probably diverge).
`eks`	switch for the Kohn-Sham energy (0 or 1). The Harris energy is always evaluated. The extra steps needed for the Kohn-Sham energy are avoided when this switch is zero. On the other hand, the forces converge faster when the switch is on.

`restart` `{rd} 0 {wr} 0 {auto} 0 {ms pos} 1 {ms ef} 0`

Control reading and writing of the restart file.

`rd`	switch (0 or 1). If 1, start the job by reading the restart file. Otherwise, start by overlapping the free-atom densities.
`wr`	switch (0 or 1). If 1, rewrite the restart file at job completion with the data from the last completed iteration.
`auto`	interval in which to autosave data in the restart file. No autosaving if zero.
`ms pos`	switch (0 or 1). If 1, use the positions from this file instead of the restart file.
`ms ef`	switch (0 or 1). If 1, use the Fermi energy window from this file instead of the restart file.

Normally, the atomic positions and the Fermi-energy window are also read from the restart file. To override this, *e.g.*, if a self-consistent density is to be used as starting guess for a different geometry, switch `ms pos` must be used. For a metal, the Fermi energy will then make a jump which can be much larger than the width of the converged Fermi energy window. To handle this properly, the window must be widened judiciously and switch `ms ef` must be used. The same precaution should be taken when other parameters are changed for a selfconsistent metal, for example the basis set.

$\boxed{\texttt{switches}}$ `{rel} 1  {spinp} 0  {friz} 0  {xc} 1  {grad} 0`

Switches to determine the type of calculation.

`rel`	switch for scalar-relativistic treatement (0 or 1).
`spinp`	switch for spinpolarization (not active yet).
`friz`	switch to freeze phi and phi-dot for better forces (0 or 1).
`xc`	switch for exchange-correlation parametrisation. Values: (1) Ceperly-Alder with the Vosko parametrisation, (2) Hedin-Lundqvist, (3) Ceperly-Alder nonspinpolarised.
`grad`	switch for gradient corrections (not active yet).

$\boxed{\texttt{lattice}}$ `{alat} 10.20  {vecs} 0 .5 .5  .5 .0 .5  .5 .5 .0`

Define the crystal lattice.

`alat`	lattice constant.
`vecs`	basis vectors for the lattice. Multiplied by the lattice constant internally. If one vector is considerably longer than the others (as for a slab calculation), it must be the third one. Under this condition the program can recognize whether additional lattice vectors are needed for accurate Ewald sums.

$\boxed{\texttt{ftmesh}}$ `{n1,n2,n3} 12 12 12  {tolgv} 1d-7`

Define the real-space mesh used for fast Fourier transforms.

`n1`	number of real-space mesh divisions along the first lattice vector. This must be a number which can be handled by the FFT routine, typically factorizable by 2, 3 and possibly 5.
`n2`	same for second lattice vector.
`n3`	same for third lattice vector.
`(tolgv)`	tolerance for orbital-dependent cutoffs in recip space.

These parameters are of primary importance for the tradeoff between accuracy and computation time.

$\boxed{\texttt{symgroup}}$ `< r3d  i r4z:t(.25,0,.25) >`

Define the symmetry operations.

`<..>`	string which defines the symmetry group generators. Each generator consists of a point operation, optionally followed by a translation. Generators are separated by spaces. Spaces

are not allowed within a generator.

Point operations:

`r`n`v`	n-fold rotation around a vector **v**
`mv`	mirror taking vector **v** into −**v**
`i`	inversion
*op*1`*`*op*2	product of any two of the above

Non-symorphic operations:

op`:tv`	*op* followed by translation **v**

Vectors can be written as follows:

`(`x`,`y`,`z`)`	explictly using coordinates
`x`	equivalent to `(1,0,0)`
`y`	equivalent to `(0,1,0)`
`z`	equivalent to `(0,0,1)`
`d`	equivalent to `(1,1,1)`

$\boxed{\texttt{kmesh}}$ `{job} 1  {nkxyz}  4 4 4`

Define the k-point mesh for the Brillouin zone integration.

`job` selects the type of mesh.

Codes:

0	standard regular mesh containing Γ
1	special points mesh for fcc
3	special points mesh for sc or tetragonal
4	special points mesh for bcc
5	special points mesh for hcp

`nkxyz` divisions along the vectors spanning the reciprocal lattice

This line is only used by the program which sets up the k points.

$\boxed{\texttt{metal}}$ `{lmet} 1  {smear} 1.025  {ef,def}  1.6   0.3`

Switch for metal treatment, and associated parameters.

`lmet` switch for metal treatment (0 or 1).

`smear` parameter for gaussian smearing of eigenvalues. The integer part is the order of the MP function, the fractional part is the smearing width (an energy).

`ef,def` center and spread of the 3-point mesh to bracket the Fermi energy. See section 5.1.2.

Default: semiconductor occupation of eigenstates.

`dos` `{ldos} 0  {window} 0  2.5   {pts} 2001`

Switch for density of states, and associated parameters.

`ldos`	switch (0 or 1). The DOS is written to `<id>.dos` if one.
`window`	the minimal and maximal energy of the window.
`pts`	the number of points used to tabulate the DOS.

Default: no density of states is written to disk.
For a metal, the DOS is always made, which means that the window must be defined sensibly even if `ldos` equals zero. For a semiconductor, the dos is only made if `ldos`=`1`. In that case, a `metal` line (with `lmet` equal to zero) is needed to define the smearing function.

`amp` `{u} 0.1`

Set the amplitude for atomic shifts.

`u`	size of shift.

Default: no shifts, `u=0`.

`pos` `<si>  {pos}  -.125 -.125 -.125   {shift} -1 -1 -1`

Define the position for one atom.

`<..>`	label for the atom species. Leading and trailing blanks within the label string are ignored. This species must be defined somewhere in the file.
`pos`	Cartesian coordinates of the atom. These are multiplied by the lattice constant internally.
`(shift)`	optional shift vector with default $(0, 0, 0)$.

Altogether, the position used is: `alat` × `(pos` + `u` × `shift)`.

`xpos` `<si>  {pos}  0.5 0 0   {shift} 1 0 0`

Define the position for one atom in terms of the lattice vectors.

`<..>`	label for the atom species.
`pos`	position parameters for the atom.
`(shift)`	optional shift vector with default $(0, 0, 0)$.

This line specifies the atom position and the optional shift vector as linear combinations of the three lattice vectors, defined in the `lattice` line. When the input file is read, the positions are transformed to cartesian coordinates internally.

Lines for Species Definitions

A species collects together data for a certain type of atom under one label. This label is referred to in a `pos` line. Each species definition is bracketed between a matching pair of `spec` and `endspec` lines.

To make things easier, default values for some species parameters can be triggered by setting the corresponding parameter equal to zero. The used values are printed to the listing near the start of the job. A species block which uses this feature extensively might look like this:

```
spec  <si>
  gen   {z} 14.0  {mass} 1  {rmt} 2.20
  chd   {lmxl} 2  {rg*} 0   {rsmv*,kv*}  0 0
  aug   {rsma*} 0  {kmax} 4  {lmax} 3  {pnu} 3.5 3.5 3.3 3.15
  foca  {lfoca} 1  {rfoca*} 0
  bas   'h' {l} 0 2 {rsm} 1.4  {e} -0.5
  bas   'h' {l} 0 1 {rsm} 2.2  {e} -1.4
  fa    {rsmfa*} 0   {Qat}  2.0  2.00
endspec
```

Parameters with this default function are marked by a star.

$\boxed{\texttt{spec}}$ `<si>`

Start the definition of a new species.

`<..>`	The label string for the species (up to eight characters). Leading and trailing blanks within the string are ignored.

$\boxed{\texttt{gen}}$ `{Z} 14.0 {mass} 28.09  {rmt} 2.20  {nr} 301  {A} 0.02`

General data for the species.

`Z`	nuclear charge.
`mass`	atomic mass (presently not used).
`rmt`	muffin-tin radius.
`(nr)`	number of points in the radial mesh (must be odd).
`(A)`	compression parameter for the radial mesh.

The radial mesh is given by $R_i = B[e^{A(i-1)} - 1]$. If `nr` and `A` are not given, the free atom program chooses the mesh. For heavier atom, more mesh points are required. Parameter `A` should be about 0.02 for non-relativistic calculations. For the scalar-relativistic case, more mesh points and a larger `A`$\approx$ 0.03 are needed. It might be worth checking that the default mesh has enough points.

`chd` `{lmxl} 2  {rg*} 0.55  {rsmv*,kv*}  0.95  15`

Parameters for the charge-density representation.

`lmxl`	angular-momentum cutoff for the local electron density. `lmxl=0` is not unreasonable but a higher cutoff might be needed, depending on the atom and the application.
`rg*`	radius of the gaussians used to fix up the multipole moments of the smooth mesh density. The gaussians must be well inside the muffin-tin sphere, typically `rg` equal to one-fourth of the MT radius. The exact value is not especially critical.
`rsmv*,kv*`	parameters for the P_{kL} expansion when overlapping the free-atom densities. The smoothing radius `rsmv` is typically one-half of the MT radius and `kmxv` is about 10. These parameters are uncritical, unless the Harris energy for the first iteration is of special interest.

`aug` `{rsma*} 0.90  {kmax} 4  {lmax} 2  {pnu} 3.5 3.5 3.3`

Parameters for the augmentation.

`rsma*`	smoothing radius for the expansion of the envelopes for augmentation. Typically one-third to one-half of the MT radius.
`kmax`	cutoff for the augmentation expansion. Typically 4 or higher.
`lmax`	angular-momentum cutoff for the augmentation. This must be at least as high as the maximal angular momentum used in the basis on this atom.
`pnu`	boundary conditions of the radial wave functions for `l=0` to lmax. The integer part is the principal quantum number and the fractional part translates to the logarithmic derivative. Normally, these are just starting values since the program will adjust the boundary conditions to the center of the occupied bands while iterating. If a value is negative, it is frozen throughout the calculation. This might be needed to supress spurious "ghost bands." For higher angular momenta, the `pnu` values approach those of free electron states (0.5, 1.25, 2.147, 3.102, 4.078, 5.063, 6.052). For small spheres, it is sometimes important start from these.

The expansion parameters `rsma`, `kmax`, and `lmax` have some significance for the accuracy and computation time, although not as much as the real-space mesh and the basis. The expansion is only needed for the smooth tails, so a low `kmax` cutoff is often sufficient. For a given `kmax`, the accuracy of the expansion can sometimes be improved by choosing `rsma` optimally.

`bas` `<h>  {lmn,lmx} 0 2  {rsm} 1.4  {eh} -0.5`

Add a block of functions to the LMTO basis.

`<h>`	string to define the type of basis functions to be added. Presently, only `<h>` for smooth Hankels is permitted.
`lmn,lmx`	angular momentum range for this block.
`rsm`	smoothing radius.
`eh`	localization energy (must be negative).

Together with the real-space FFT mesh, the LMTO basis is of high significance for the computational effort. Each `bas` line adds functions from the lower to upper `l` limit. An optimal basis might need different values of `rsm` and `e` for the different angular momenta.

`fa` `{rsmfa*} 1.25   {Qat} 2.0  2.0`

Parameters for the free-atom calculation.

`rsmfa*`	smoothing radius of Hankels used to fit the free-atom density, typically one-half of the muffin-tin radius.
`Qat`	s, p,... occupation numbers for the free atom.

`foca` `{lfoca} 1  {rfoca*} 1.3`

Switch for FOCA, and associated parameters.

`lfoca`	switch (0,1 or 2). When nonzero, the frozen overlapped core approximation is used on this atom. For 1, the smoothed core density is represented on the real-space mesh. For 2, the exchange-correlation term is done to first oder in the density (this is more experimental).
`rfoca*`	Smoothing radius, used to represent the Hankel fit to the tail density on the mesh. See section on FOCA.

Default: foca not used on this atom, `lfoca=0`.

`endspec`

Closes the definition for the current species.

Special-Purpose Lines

These are additional lines to be placed outside a species block, used to set less-frequently used parameters or when generating data for a plot.

```
ewald   {a} 2  {tol} 1e-12   {nkdmx,qmx} 800 800
```

Parameters for the Ewald summation.

`a`	dimensionless Ewald parameter, automatically scaled for a suitable partitioning in most systems.
`tol`	tolerance for Ewald sums. The cutoff radii are chosen so that the truncated terms are smaller than `tol`.
`nkdmx,qmx`	Space allocated for the real respectively recip vectors.

Default: some generally reliable setting.
Ewald summation splits lattice sums into separate terms done in real and recip space. If `a` is increased, fewer real-space and more recip-space terms are used. The main purpose of this line is to reduce the tolerance if the Ewald summation is suspect.

```
stretch   {vx,vy,vz} 0 0 1  {gam} 1.05
```

Apply a volume-conserving stretch to the unit cell.

`vx,vy,vz`	Vector giving the direction of stretch.
`gam`	Magnitude of the stretch. Directions parallel to the vector are multiplied by γ, orthogonal directions by $1/\sqrt{\gamma}$.

Default: no stretch distortion, `gam=1.0`.
This line helps to calculate elastic constants. Vectors appearing in the input file are transformed by the distortion. Reciprocal-space vectors are transformed by the inverse operation. The strain tensor for the distortion is printed in the listing. The printed shears are the tensor quantities [18] and not the engineering strains (multiplied by two).

```
symln   {nq} 15  {q1} 0.0 0.0 0.0  {q2} 1.0 0.0 0.0
```

Define a symmetry line for a band-structure plot.

`nq`	Number of points along the symmetry line.
`q1`	starting point.
`q2`	end point.

This input is only needed when plotting the band structure. One line is used for each panel in the plot.

$\boxed{\texttt{plane}}$ `{vx} 1 0 0    {vy} 0 1 0    {center} 0 0 0`

Define the plane for a charge-density plot.

`vx`	vector which defines the x direction in the plane.
`vy`	vector which defines the y direction in the plane. It is automatically orthogonalized on `vx`.
`(center)`	the origin of the plane. Default is `(0,0,0)`.

This input is only needed for a charge density plot.

$\boxed{\texttt{dplot}}$ `{x} 0.0 1.0    {y} 0 1.414   {nx,ny} 31 41`

Parameters for a charge-density plot.

`x1,x2`	limits in `vx` direction defined in the `plane` line.
`y1,y2`	limits in `vy` direction defined in the `plane` line.
`nx,ny`	number of grid points in direction of `vx` and `vy`.

This input is only needed for a charge density plot.

$\boxed{\texttt{plot}}$ `{b&s} 1  {fa} 1  {basis} 1`

Switches for auxillary plot data

`b&s`	if 1, `nchk` writes a ball-and-sticks file `<id>.bs`.
`fa`	if 1, `nfa` writes the free-atom density and the fit to it to files `<id>.pl1`, `<id>.pl2` . . .
`basis`	if 1, `nchk` writes the smooth-Hankel envelopes to files `<id>.pl1`, `<id>.pl2` . . .

Default: No files with auxillary plot data are written.

4.4 Setting up a Calculation

This section discusses the issues which have to be decided when setting up a calculation for a new system. Typical points are how to choose the sphere radii, which states are to be treated as valence, and the definition of the basis set. The different points are presented in the sequence in which they might be handled in practice.

4.4.1 Setting Up the Geometry

The first step is to place the atoms in the proper positions. This is in principle straightforward: the basis vectors for the crystal lattice, the lattice constant, and the coordinates of the atoms must be entered into the input file using cartesian coordinates. Note that the position of an atom used internally is actually the shortest equivalent vector in the unit cell.

At this point, it is advisable to ensure that the input file is formally correct in order to permit usage of the checking program, `nchk`. This means that all required input lines must be present and that each species appearing in a `pos` line must be defined. At this stage, many input parameters can be initialized with inspired guesses. The auxillary program `nchk` first reads in the input file, checks the syntax, and controls as far as possible that the various parameters have been given reasonable values. After that, a number of other steps are performed, including a check of the numerical accuracy, the symmetry operations, and the sphere overlaps.

As a first application, `nchk` prints out the interatomic distances and the bond angles. This is useful to control the geometry. For this printout, atoms are considered nearest neighbors if their distance is somewhat larger than the sum of their muffin-tin radii. If the latter are very small, expected lines will be missing in the output.

Making supercells – One task which turns up frequently is to set up a supercell based on a given lattice. To help in this step, the auxillary program `nsup` can be used. This program reads an existing input file, asks for the lattice vectors of the supercell, and calculates all atomic positions which lie within it. For convenience, the results are placed in file `<id>.tmp` as well as the listing. For example, assume we are interested in phonons at the X point for Si. An existing input file `si.mas` defines the structure using these lines:

```
lattice   {a} 10.2  {vecs} 0 .5 .5    .5 .0 .5   .5 .5 .0
pos  <si>      0   0   0
pos  <si>    .25 .25 .25
```

When requested by `nsup`, the supercell lattice vectors are entered as

```
   .5 .5 0  .5 -.5 0  0 0 1
```

The result is four sites within the supercell:

```
 pos  <si>    .000000    .000000    .000000
 pos  <si>    .500000    .000000    .500000
 pos  <si>    .250000    .250000    .250000
 pos  <si>    .250000   -.250000    .750000
```

It is the user's responsibilty to enter a supercell which is compatible with the given lattice.

4.4.2 Determining the Symmetry Operations

The most important reason to use the largest possible symmetry group is to reduce the number of k points needed for the Brillouin zone integration. In addition, systems can sometimes be converged more easily if the symmetry is higher, because there are fewer degrees of freedom for the charge to "slosh about." However, the program will also run properly, albeit inefficiently, if the set of symmetry operations is smaller than optimal.

A symmetry operation is specified as a point operation, optionally followed by a translation. The complete symmetry group is defined by entering specific elements of the group g_1, g_2 ... as generators (see the description of the `symgroup` line). The group consists of all distinct finite products which can be assembled out of the generators. For example, the full 48-element cubic group is generated by the three point operations `< r3d r4z mx >` and a 16-element tetrahedral group is made using `< r4x mx mz >`.

Assuming the symmetry group is not known beforehand, the first step is to find specific operations which map the crystal into itself. It is usually not a problem to guess point operations which are compatible with the underlying lattice. However, it can be more difficult to find translation vectors which result in a valid group operation when combined with the point operation. For this purpose, the auxillary "symmetry finder" program `nsymf` is available. This program takes one generator in the `symgroup` line at a time, ignores any translational part specified there, and tries to find a translation which makes a valid symmetry operation.

Returning to the example of the silicon X phonons, assume we have in the meantime made an input file for the TO(X) phonon containing these lines:

```
lattice   {a} alat  {vecs} 0.5 0.5 0  0.5 -0.5 0  0 0 1
amp 0.01
pos  <si>    .000000    .000000    .000000   1  1  0
pos  <si>    .500000    .000000    .500000  -1 -1  0
pos  <si>    .250000    .250000    .250000   1  1  0
pos  <si>    .250000   -.250000    .750000  -1 -1  0
```

Note that the atomic positions have been shifted slightly in a way which defines the phonon of interest. As point operations which might make valid group elements when combined with suitable translations, we enter these candidates in the `symgroup` line:

```
symgroup  < r4z m(1,-1,0) mz  i r2z >
```

The point operations used in this step must be chosen to be compatible with the lattice spanned in the `lattice` line; the program will not detect if this is not the case. Program `nsymf` simply tries each vector connecting two atoms as the translation vector, checking whether the resulting operation maps each position into one occupied by the same type of atom. For large cells, this might take a while. Running `nsymf` for the present example generates this output:

```
-------  operation r4z  -------
**** no translations found ****
-------  operation m(1,-1,0)  -------
valid translations are:
    .000000     .000000     .000000
-------  operation mz  -------
valid translations are:
   -.250000    -.250000     .250000
-------  operation i  -------
valid translations are:
    .250000    -.250000    -.250000
-------  operation r2z  -------
valid translations are:
    .500000     .000000    -.500000
```

This shows that the `r4z` cannot be used, while the other cases make valid generators when used with the displayed translations.

The next task is to set up the symmetry group from these generators. This is one of the steps done in `nchk`. If all the valid generators above are included, `nchk` produces an output similar to the following:

```
----- Check symmetry -----
Set up symmetry group:
symgrp:  m(1,-1,0) mz:t(-.25,-.25,.25) i:t(.25,-.25,-.25)
         r2z:t(.5,0,-.5)
```

```
order of generator  1  is    2
order of generator  2  is    2
order of generator  3  is    2
generator  4  is already in group as element  7
number of elements in space group is   8
```

Evidently the fourth generator is redundant and should be left away, since it is already in the group created by the first three. The final group has eight elements. Next, the program splits the atoms into symmetry classes (or displays an error message if an operation does not map an atom into a similar one):

```
Number of symmetry classes:    1
Symmetry class   1:
atom   ib   species              position
   1    1   si         .010000    .010000    .000000
   2    2   si        -.010000    .490000   -.500000
   3    3   si        -.240000   -.240000    .250000
   4    4   si        -.260000    .240000   -.250000
```

Two atoms are in the same symmetry class if there is at least one operation taking one into the other. In the present case, all atoms are in the same class, but this will not necessarily be the case for other systems. The size of the symmetry group and the symmetry classes should be inspected; if they are unreasonable, presumably some operations are still missing in the group, and additional generators should be sought.

4.4.3 Choosing the Muffin-Tin Spheres

The next important question is how to partition space into muffin-tin spheres and an interstitial region. In principle, this partition should only influence the convergence of the different representations but not the results which come out. However, in practice this is not strictly true. For example, when the core states begin to have a non-negligibe amplitude at the sphere surface, their contribution to the total energy can depend sensitively on the MT radius.

The following points influence the choice of the MT spheres:

- As a general rule, a set of muffin-tin radii should be chosen in the beginning and then kept fixed as far as possible. Specifically, when searching for the equilibrium lattice constant it is better not to scale the radii. When the atoms are moved about, *e.g.*, for a frozen phonon, the muffin-tin radii should also be kept fixed. Usually this means that the spheres are chosen a some percent smaller than overlapping in the unperturbed system.

- In view of the preceeding point, the muffin-tin spheres should be as large as possible. This is a question of program efficiency: the steps with a large numerical effort are concerned with the interstitial region, and will be more efficient if it is smaller.
- States which are assigned to the core should, if possible, be concentrated within the MT sphere. Often, however, this will not be possible. The frozen-overlapped core approximation (FOCA) should help in these cases.
- The ϕ and $\dot{\phi}$ functions used within the sphere should be "adequate as a basis" for the valence wavefunctions. This is somewhat vague, of course. For systems such as SiC the results depend somwhat on whether the two spheres are equal or whether the C sphere is chosen smaller than Si. The reason is that the quality of the basis inside the spheres changes.

For a homogeneous system with one type of atom, the obvious prescription is simply to make the (identical) spheres as large as possible, then to shrink them a few percent to allow atomic shifts. It is for compounds that the real work begins. Often, the best choice is based on a healthy dose of chemical intuition. To a large extent, the idea is that the radius of an atom is a reasonably well-defined and transferable quantity, which can be obtained by looking at various simpler systems. A simple example would be PdH. Chemically, the picture is that of an fcc Pd lattice with H atoms placed in the interstitial sites. A good choice is to set the Pd radius to the largest value which avoids overlap in the pure Pd crystal. Second, the H spheres are chosen to fill out the interstitial holes in the PdH structure. As a cross check, this gives a radius in the order of one-half of the bond length of an H_2 dimer.

The assumption of well-defined and transferable sphere radii is reasonably reliable as long as charge transfer does not play an exceptionally large role. For systems with large charge transfer (such as many oxides) knowledge of the expected ionic state can be helpful. A valence wavefunction typically exhibits a few oscillations because of othogonality to the core states, then reaches a final maximum before decaying at larger distances. Generally, the MT radius should lie somewhat outside this last maximum. If an atom has gained electrons by charge transfer, this maximum is pushed to larger radii and the atom is effectively bigger. Conversely, an atom which has lost electrons is effectively smaller. However, this effect should not be overestimated. Even in a system such as NaCl, the amount of charge inside a muffin-tin sphere is not that much different for the free atoms and the crystal.

4.4.4 Defining the Valence States

The next important choice is to separate the electron states into valence and core states. In the present version of this program, only one principal quantum number can be included as valence state for each angular momentum. If a principal quantum number is known to be partly filled in the crystal, it is relatively clear that this should be treated as valence. More or less equivalently, orbitals which play a relevant role in the chemical bonding are valence states. In unproblematic situations, the states of the same angular momentum but smaller principal quantum numbers will then be suitable for treatement as core states. In the first instance this means that essentially all of the core charge lies within the muffin-tin sphere. Examples are Si with the configuration $3s3p$ and most late transition metals, *e.g.* Pd with the configuration $5s5p4d$.

Things are more complicated when, for a given angular momentum, one principal quantum number is associated with a full band some way below the Fermi energy, whereas the next-higher principal quantum number lies above it. Then it is an open question whether it is more important to treat the low-lying band within the valence states, or to handle it as core state to permit inclusion of the higher quantum number for more flexibility in the valence states. Often, the higher-lying states turn out to be quite important, especially for the energy changes when atoms are shifted. As an example, in Ga the $3d$ shell is full but not very deep. For phonons in compounds such as GaAs, it is more important to include the $4d$ states as valence. Hopefully, in these cases the FOCA will prove to be the solution.

Systems which have problems in this context are the early transition metals, say Ti with the atomic configuration $4s^2 3d^2$. The $3p$ shell was completed in Ar which is only four atoms further back. Therefore the $3p$ states spill out of the MT sphere to some extent and make a reasonably wide band. On the other hand, experience has shown that the $4p$ degrees of freedom are important for the properties of the metal. Based on current experience, it seems that the FOCA (here applied to the $3p$ states, with $4p$ in the valence) is a good solution for such a case.

The next question is which states of higher angular momenta to include as auxillary degrees of freedom. For example, the nominal valence configuration of Si is $3s^2 3p^2$ but a calculation will generally also include the $3d$ states in the basis. Formally, this depends on how much higher the prospective auxiallary states lie in energy compared to the nominal valence states. Most atoms typically use at an *spd* basis, although a restriction to *sp* orbitals can work in the row containing B, C, and N.

Finally, it is again systems with large charge transfer which cause extra

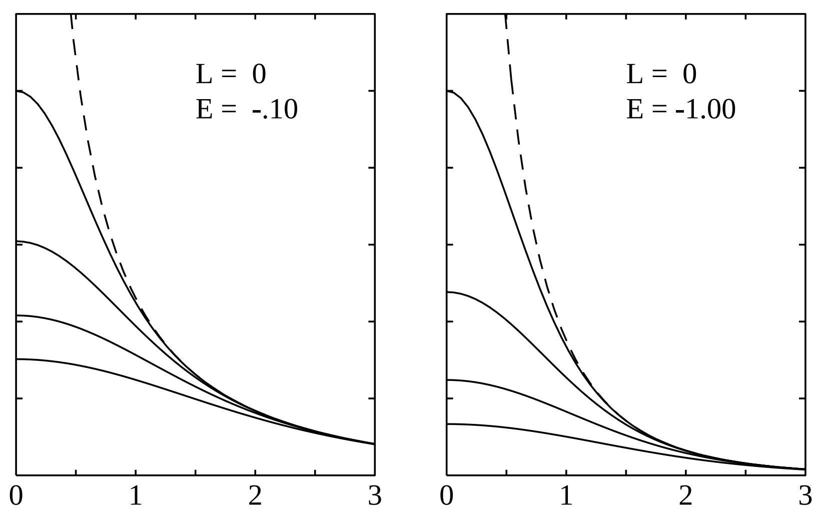

Figure 4.2: Plots showing how a smooth Hankel function changes when the smoothing radius is varied. Shown here are the s functions (l=0) with energy -0.1 (left panel) and -1.0 (right panel). The curves have the smoothing radii 0.6, 0.9, 1.2, and 1.5 (from top to bottom). The standard unsmoothed Hankel functions are shown dashed. The vertical scale is arbitrary.

difficulties. Charge transfer to an atom leads to a repulsive potential which can shift the energies of the different orbitals by some amount. Thus, previously inert core states can shift up to the valence region while other orbitals might be pushed out of the range of interest. The opposite effect happens when an atom loses charge. It might be advantageous to develop a clear picture of which states hybridize to form the band structure in the finished compound before deciding on the final valence orbitals.

4.4.5 Choosing the Basis Set

Next, the basis for each species must be chosen. The shape of each basis function is determined by two parameters. The (negative) energy parameter determines how fast the orbital decays at large radii, and the smoothing radius determines how strongly the function bends over near the central site. Plots for some typical parameters are shown in Figs. 4.2, 4.3, and 4.4. Generally the smoothing radius $R_{\rm sm}$ is in the range from $0.5R_{\rm mt}$ to $R_{\rm mt}$ and the energy parameter ϵ is between -0.1 and -2.5 Ry. From the plots, it is readily appreciated that the smoother functions can be integrated using a coarser mesh. Note also that the d functions bend over more strongly, suggesting that smaller smoothing radii are appropriate for d functions.

Frequently the first calculations for some new topic will be for a small

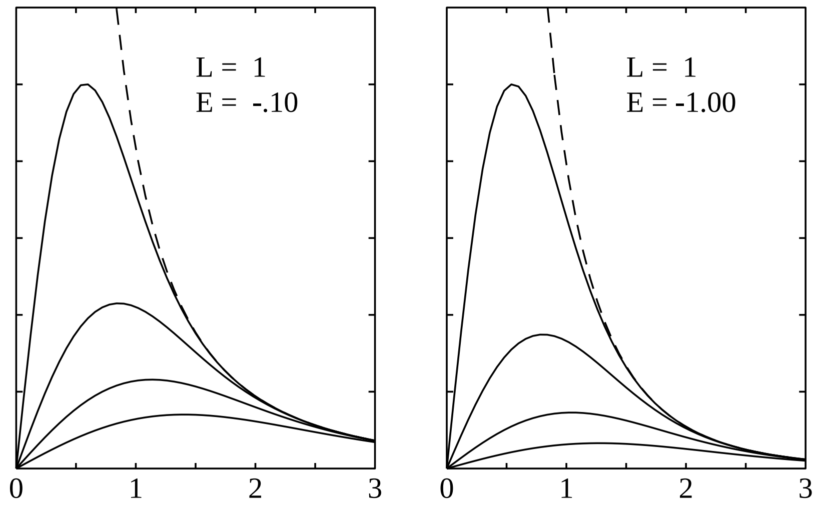

Figure 4.3: Same as the Figure 4.2, but for p functions ($l = 1$).

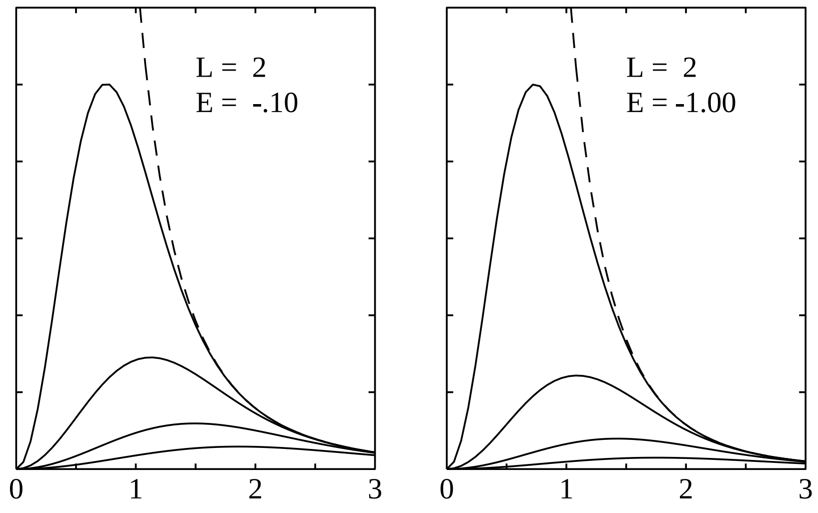

Figure 4.4: Same as the Figure 4.2, but for d functions ($l = 2$).

trial system, perhaps containing up to three atoms in the unit cell. In that case, it is a good idea *not* to worry about using an optimal basis in the beginning. Instead, it is better to simply stock up the basis until adequate convergence is reached. One (not especially good) possibility is to simulate the standard unsmoothed functions, by using smoothing radii of about one-third to one-half of the muffin-tin radius. The functions would then be distinguished by their energies. Be aware that an expensive real-space mesh will be needed for this (see the check of the numerical accuracy below). Alternatively, it is probably better to start with functions of some reasonable smoothing radii (something like 70% of R_{mt} for s and p orbials, somewhat smaller for d states). After that other functions are added to increase the degrees of freedom, chosen with significantly different R_{sm} or ϵ to ensure numerical stability. For a slab, all energy parameters should be more negative than about -0.7 Ry in order to describe the exponential decay of the eigenfunctions into the vacuum.

As the basis set is increased, the total energy should rapidly approach a limit from above. It is usually good enough to consider only the Harris energy of the first iteration for these tests. The relevant variational quantity is here the sum of the eigenvalues in a fixed potential as function of the basis. Therfore, we seek a minimum of the Harris energy in this context. After a converged basis set is found, some self-consistent energies should be calculated. For example, a good standard test is the the binding-energy curve (the energy as function of the lattice constant) which gives the equilibrium lattice constant and the bulk modulus. It can also supply the binding energy, but for this purpose the spin-polarization energy of the free atom must be obtained somehow.

Somewhere along the way, it will be desirable to optimize the basis properly. The aim is to tune the parameters R_{sm} and ϵ for each basis function to obtain a basis set which is close to convergence but as small as possible. This is considered next.

When to optimize the basis – To begin with, the basis does not necessarily have to be optimized. If the system of interest is small and the calculations are fast enough using the *ad hoc* converged basis, the basis can be kept as it is throughout.

Secondly, the optimisation step can be avoided if an input file is available for a previous calculation for a similar system. Presumably a properly determined basis can be transfered between different environments. An explicit test is still advisable, of course. For example, it can be checked that the binding curve does not shift significantly when some additional functions are added.

A good reason to optimize the basis is if the real system of interest contains more than two or three atoms. There are two separate issues here. First, computational efficiency is important for a large system. Second, possible stability problems due to a nearly overcomplete basis set become more severe. If the small test system is representative enough, the basis optimisation should for this and the result transfered to the larger system later.

How to optimize the basis – It is not recommended to search directly for the formal total-energy minimum in the multi-dimensional space spanned by the basis parameters. The result will probably be two identical functions for some angular momentum, since (for symmetry reasons) the energy is stationary there. Instead of choosing functions with almost identical parameters, it is much more stable and only insignificantly higher in energy to use basis functions with substantially different shapes.

To present experience, a good procedure is essentially as follows. First, a single optimal function is determined for each angular momentum. After a minimal basis set [1] has been constructed in this way, further functions are added until adequate convergence is reached. The shape of the additional functions is less critical, so they can be chosen with substantially different parameters.

To do this in practice, a minimal basis set is guessed as starting point. For one angular momentum at a time, the Harris energy in the first iteration is then minimised respective to the corresponding smoothing radius and energy parameter. It is good enough to vary R_{sm} first, then the energy parameter, then to go on to the next angular momentum without repeating the procedure for this one. The aim is not to find the exact minimum, but to find ballpark parameters which mimic the shape of the true wavefunctions. Sometimes the energy will decrease slowly but continually as R_{sm} is made larger. In that case, choose R_{sm} somewhat smaller than R_{mt}. For a given geometry, it is not unusual that this optimisation produces a minimal basis with a total energy only a few mRy above the converged value. Nevertheless, in order to make a basis which can be used reliably in different environments, some additional functions will probably be needed for those angular components which play a role in the chemical bonding.

The final criterium is that the self-consistent energies are insensitive to an additional stock-up of the basis. Therefore, a final check of this nature should be done, possibly for the binding curve or a simple phonon shift. In many cases a reasonably small rigid shift of the energy surface can be tolerated, since equilibrium geometries and phonon frequencies are not affected by this.

[1]A minimal basis set has one function per angular momentum.

All other things being equal, one might as well chose the R_{sm} values larger since this will permit a coarser real-space mesh. Also, L channels which turn out to have similar parameters can be combined to one block, which can make program execution slightly faster. In the end, the smallest smoothing radius used on any species will limit the mesh. The whole optimisation procedure should be done with a mesh fine enough to guarantee numerical accuracy throughout (see below). The final step is to make the mesh as coarse as possible for the chosen basis. Similarily, the augmentation cutoff should first be set high enough to ensure accuracy for all considered functions. In the end, it can be tuned to be small, although this is not as relevant for the efficiency compared to the real-space mesh.

4.4.6 Checking the Numerical Accuracy

Two different issues are central here. The first is the real-space mesh needed for an accurate representation of the smooth-Hankel envelope functions. The second concerns the parameters of the local expansion for the augmentation. For the computation time, the size of the real-space mesh is the most important parameter. A measure for the accuracy in the two steps can be obtained using `nchk`. However, it is strongly suggested to do a direct test for a prototype system sometime, whereby the eigenvalues and the total energy are inspected directly as the parameters are changed. As usual, such a test can be done without self-consistency iterations, either for the the overlapped free atoms of for a self-consistent potential.[2]

Below, the mesh and the augmentation expansion are discussed separately, followed by some notes on the remaining parameters which must be set.

Real-Space Mesh

When chosing the real-space mesh, the number of divisions along the edges of the unit cell (set in the `ftmesh` line) should define a mesh which is reasonably homogeneous. For example, using 12 divisions for each direction in the Si TO(X) cell leads to this printout in `nchk`:

```
gvcutoff: real-space mesh has   12  *  12  *  12   divisions
mesh is spanned by vectors of length     .601   .601   .850
```

[2]When starting from a self-consistent calculation for a metal, the Fermi-energy window must presumably be widened before the numerical parameters are changed. See the section on Brillouin-zone integration for metals.

This mesh is inefficient because the spacing in one direction is substantially larger than in the others. A better choice is to use 16 divisions in the longer direction:

```
gvcutoff: real-space mesh has   12  *  12  *  16   divisions
mesh is spanned by vectors of length     .601   .601   .637
```

For the selected real-space mesh, the program chooses a cutoff radius in reciprocal space and represents mesh functions as a Fourier sum over the reciprocal vectors inside the cutoff sphere. We would like to know whether this mesh is fine enough to evaluate integrals over the unaugmented envelope functions. As a measure for the accuracy, `nchk` evaluates the the diagonal matrix elements for the overlap and the kinetic energy by mesh integration and compares the result to the analytical answer. The results are presented in a block similar to the following:

```
----- Check real-space mesh accuracy -----

Real-space mesh divisions:   12   12   16
Integration errors for diagonal matrix elements:

species   block    rsm      e      l      overlap    kinetic energy

   si       1     1.400   -.500    s      5.46D-13      3.03D-11
                                   p      3.04D-11      6.33D-10
                                   d      8.20D-10      8.87D-09
   si       2     2.200  -1.400    s      1.55D-14      8.24D-14
                                   p      1.24D-13      5.34D-13
```

In this example, the basis consists of an *spd* block of smoothing radius 1.4 and energy -0.5 and an *sp* block with smoothing radius 2.2 and energy -1.4. For each angular momentum the relative errors in the numerical integrals are shown. It can be seen that the accuracy depends sensitively on the smoothing radius: for the larger smoothing of 2.2, the error is about three orders of magnitude smaller. To present experience, an accuracy of at least 10^{-6} to 10^{-8} is desirable, showing that the mesh can be somewhat coarser in the present case.

A note: the “exact” result used for comparison is obtained using Ewald summation, which has a small error itself. For very fine meshes, the test described here might make it necessary to reduce the tolerance in the `ewald` line.

Augmentation Expansion

In the augmentation step, each envelope tail is expanded as a sum of the polynomials P_{kL} up to a prescribed angular momentum cutoff $L_{\max}^{\mathrm{A}}$

and maximal polynomial order $k_{\max}$. The polynomials are defined by the smoothing radius $R^{\mathrm{A}}_{\mathrm{sm}}$, which is a measure for the range over which the expansion tries to fit the function.

In the listing from `nchk`, the accuracy of the augmentation expansion is estimated by a block similar to the following:

```
----- Check accuracy of augmentation expansion -----

RMS errors in expanded head and tail functions:

Species si:   rmt= 2.2000   rsma= 1.2500
augment up to  lmax= 2   kmax=  4

block    rsm        e      l      head        tail

  1     1.400    -.500     s    6.07D-03    7.83D-07
                           p    1.21D-02    2.52D-07
                           d    1.85D-02    1.77D-07

  2     2.200   -1.400     s    4.71D-04    1.21D-04
                           p    6.88D-04    4.30D-05
```

Here, the "head" is the smooth Hankel function on the central sphere. The "tail" is the Bessel function for the same energy and angular momentum, which is one component of the envelope function in a sphere some distance from the center. Both of these functions are expanded as a sum of the P_{kL} poynomials, using the prescribed smoothing radius `rsma` and cutoff `kmax`. The resulting approximation is compared to the exact function and the RMS error is printed in the table. The errors for the tail expansion are a good measure for the accuracy, but they do not cover all cases used later on.

The column with the heads is for historical reasons and for orientation only, since this expansion is not used in the calculation. Attention should be focused on the tail expansions, since these are used when making the secular matrices. To present experience, errors should be approximately 10^{-4} or smaller. Of course, the accuracy is increased when `kmax` is made larger. For a given `kmax`, it is instructive to vary the smoothing radius and see how the error changes. The minimal error is generally when `rsma` is somewhere between one-third to one-half of the muffin-tin radius.

Other Parameters

A look at a typical input file shows that some further parameters must still be set. Several of these are true convergence parameters: either a user has developed a good feeling for reasonable values, or they should be increased until acceptable convergence in the final results is reached. By applying the second procedure, the first will in time become feasible.

- The angular-momentum cutoff for the local representation of the density, `lmxl`. As a starting value, zero can be used. Depending on the application and the required accuracy, higher values (until perhaps four) might be needed. Additional experience is needed for a more exact recommendation.
- The angular-momentum cutoff for the augmentation, `lmxa`. This is a convergence parameter. It must be at least as high as the maximal angular momentum used in a head function. Typical values are 2 or 3.
- The width `rg` for the gaussians to fix up the moments of the smooth pseudodensity. This seems to be uncritical, a typical value is one-fourth of the muffin-tin radius.
- Parameters for the Brillouin-zone integration – see corresponding section.
- The parameters used to expand the free-atom density around the atomic sites (`rsmv` and `kmxv`). These are usually uncritical, typically one-third to one-half of the muffin-tin radius for `rsmv` and 8-10 for `kmxv`.

4.5 Running a Calculation

This section discusses the steps done in a typical calculation, whereby attention is drawn to those output values in the program listing which should be given some attention. Most of these values are also written to the `<id>.log` file. Job control using environment variables is also discussed.

4.5.1 Basic Sequence

To run a program, environment variable `LMJOB` must first be set to the identifier for the input and output files. That is, if the input file is `si.mas` and the output files are `si.log`, `si.rst` and so on, enter the line

```
setenv LMJOB si
```

(assuming the C shell is being used). After setting up a valid input file, the standard way to proceed is as follows:

1. Run `nchk` to perform various checks on the data.

2. Run `nfa` to make the free atoms selfconsistent.

3. Run `nqp` to set up the k points for the Brillouin-zone integration.

4. Run `nfp` to perform the main self-consistency iterations.

The last step can be repeated to do additional iterations if the restart file `<id>.rst` was written at the end of program `nfp`.

Some notes concerning these steps follow. Their purpose is to point out which numbers to look at in the output.

Program nchk – First of all, the input file is read in, the syntax is checked, and a sanity check is performed on the parameters. If the syntax is incorrect, it is advisable to set variable `trace` in the input file to help find the error. The program also prints out various relevant quantities, such as the dimension of the secular matrices. The other functions were discussed in the previous section.

Program nfa – This program makes the free atoms selfconsistent and writes the density, the potential, and other data to file `<id>.atm`. The most frequent mistake which can happen here is that the atoms are not neutral, either because the principal quantum numbers (taken from the `pnu`'s in the `aug line`) are wrong or because the occupations (taken from the `fa` line) are incorrect. Thus, make sure the printed "valence charge transfer" is zero. After the atom is made selfconsistent and the total energy terms are printed, a table similar to the following is displayed:

```
Charges:      inside      outside       sum
valence     1.875967     2.124033     4.000000
core        9.998702      .001298    10.000000
total      11.874669     2.125331    14.000000
```

This is useful to decide on the treatment of the core states, since it shows how much of the free-atom core charge spills out of the muffin-tin sphere.

The next step is that the free-atom valence density is fitted to a sum of smooth Hankel functions outside the muffin-tin radius. This fit is only used when overlapping the free-atom densities in `nfp` to start up a calculation. Thus, problems hereby do not matter in the end, although they might persist for a while as the starting density is gradually diluted by the mixing procedure. Of course, when the Harris energy for the first iteration is of special interest, the fit to the free-atom density must be accurate. The fit writes reasonably self-explanatory output to the listing:

```
rhoat_fit: fit valence density outside rmt= 2.20000
use  6 smoothed hankels, rsm= 1.25000
  E:    -1.0000    -2.0000    -4.0000    -6.0000    -9.0000   -15.0000
  C:  4.052D-01  5.168D+00  6.801D+01 -6.742D+02  3.377D+03 -2.153D+04
       r          rho         fit        diff
    .000016   10.439435    .129998   10.309437
   2.200000     .027876    .027905    -.000028
   2.969873     .009272    .009275    -.000003
   4.009093     .001929    .001930    -.000001
   5.411894     .000247    .000246     .000001
   7.305476     .000018    .000018     .000000
fit:  qtot  4.600540   out  2.124033   in  2.476506   rms err=   .000255
rho:  qtot  4.000000   out  2.124033   in  1.875967
```

If a larger smoothing radius is used, the quality of the fit generally deteriorates but a coarser real-space mesh can be used to represent the overlapped density. Fitting is done under the constraint that the total charge outside the muffin-tin sphere is reproduced exactly. The charge inside the sphere is not relevant, but it should not be wildly different for the fit and the true density. The table also shows the density and fit near the center of the sphere, because it is important that the fit is positive there. If not, the fitting is done a second time under the additional constraint that the total charge inside the sphere is reproduced. This may or may not help to make the density positive. By adjusting the smoothing radius `rsmfa`, a suitable compromise can usually be found.

Finally, the spilled-out tail of the core density is fitted by a single Hankel function with l=0 and an optimally chosen energy. This fit is only used in the FOCA. The relevant part of the listing is as follows:

```
coretail:  rho(rmt)=   .00138422   charge=   .00129786
fit with hankel,   e=   -31.16310   coeff=   656.61884
     r           rhoc          fit
   2.200000     .00669964     .00669964
   2.267016     .00474615     .00474908
   2.633982     .00071170     .00071137
   3.060335     .00007743     .00007649
   3.555687     .00000582     .00000560
   4.131204     .00000029     .00000026
```

In this case, the fit is done by a function of energy −31.163 multiplied by the coefficient 656.618. The fit and the exact core density are compared for different radii in the table. If the FOCA will be used, it should be checked that the fit is reasonably accurate.

Program nqp – This program uses the parameters from the `kmesh` line to set up the k-space mesh. The first step is to set up the symmetry group. In this context, the translations are ignored and the inversion element is added, so that the group printed here can be slightly different from that used in other steps. For a special-points mesh (when job is nonzero), the resulting k points are listed. The sum of the weights must alway be 2.0; the program stops if this is not the case. Finally, it can be of interest to look at the total number of k points in the Brillouin zone when symmetry equivalents are not discarded.

Program nfp – This is the central program of the calculation. Near the beginning, the various program switches are printed, controlling, for example, the scalar-relativistic treatement or the exchange-correlation potential parametrisation. The first real step is to either overlap the free-atom densities or to read in the density from a previous iteration, depending on the `read` switch in the `restart` line (see Figure 4.1). Within the main iteration loop, first the potential is made, the corresponding total-energy terms are evaluated, and the augmentation matrices are calculated. The potential parameters are printed in the standard LMTO fashion, which can give some information to those who know how to interpret these numbers. The total-energy terms are displayed as follows:

```
Energy terms:          smooth           local           total
  rhoval*vef        -28.620612      -13.956966      -42.577578
  rhoval*ves          3.901593      -31.808860      -27.907267
  psnuc*ves         115.699243    -9030.637775    -8914.938533
  utot               59.800418    -4531.223318    -4471.422900
  rho*exc           -11.612368     -147.083930     -158.696298
  rho*vxc           -15.156703     -194.499327     -209.656030
  valence chg        18.401488       -2.401468       16.000020
  core charge        40.000000         .000000       40.000000
```

As described in the description of the method, all quantities are represented as a smooth function extending through the unit cell, together with local terms added inside the various muffin-tin spheres. The corresponding contributions to the energy integrals are shown in the table. Also displayed are the valence and core charges. If these do not add up to the correct value, some numerical step is not executing properly.

Inside the loop over the k points, the eigenvalues are printed for the first point. These should definitely be checked to have reasonable values. For example, “ghost bands” normally are indicated by unexpectedly low-lying eigenvalues..

After the k loop is completed, an important step for metals is that the 3-point interpolation to the new Fermi energy is performed. It should be monitored that the Fermi energy lies within the window and that the window becomes progressively more narrow. Furthermore, the eigenvalue sum and the Fermi energy are calculated a second time using the density of states (DOS). When the 3-point Fermi energy window is still very wide, the values obtained from the interpolation and the DOS do not necessarily agree, but later on they should be nearly identical.

The next step worthy of attention (for metals as well as semiconductors) is the calculation of the new boundary conditions $P_{\nu L}$, done by placing the energy of the radial solution into the center of mass of the occupied part of the l-decomposed DOS:

```
Make new boundary conditions for phi,phidot..
site    1   species   1:si
 l  idmod    ql        ebar       pold       ptry       pnew
 0    0    .954231   -.440012   3.500000   3.783332   3.783332
 1    0   1.314145   -.286345   3.500000   3.544587   3.544587
 2    0    .102189   -.527444   3.300000   3.199885   3.199885
```

Here `idmod` is the switch determining whether a `pnu` is frozen or not, `ql` is the partial charge for this angular momentum, and `ebar` is the center-of-mass for the partial DOS. Then, `pold` is the last value used, `ptry` is the value which corresponds to `ebar`, and `pnew` is the value selected for the next iteration. In this printout, the partial charges should correspond to expectations based on the system and the `pnu` values should converge to reasonable values in time.

Next, the complete set of steps used to calculate the total-energy integrals for the input density are repeated for the output density in order to evaluate the Kohn-Sham energy. This part is completed with another block starting with `Energy terms:`... After that, the forces are printed. These should converge to stable values with iteration number, whereby the forces based on the Kohn-Sham integrals (shown in the second line for each atom) generally converge considerably faster.

Finally, the input and output density are mixed to generate a new input density for the next iteration. This is done separately for the smooth mesh density and the different local terms. Corresponding measures for the RMS difference between the input and output density are printed. These should approach zero with increasing iteration number. In fact, it is worth the effort to try different values of the mixing parameter `beta` in order to find a recipe which converges as quickly as possible. In general terms, `beta` is probably too large if quantities such as the local charges start to oscillate.

At the end of the iteration loop, the Harris and Kohn-Sham energies are printed. These should bracket the self-consistent total energy from below and above and should approach a common value with successive iterations.

Once all the requested iterations have been completed, a new restart file is written if the `write` switch in the `restart` line is set to one.

4.5.2 Job Control

The extended syntax described in Section 4.3.2 is useful when runing a series of related jobs. By using a `job` ... `endjob` block, the data and the commands for running a calculation can be combined in one file. Furthermore, by using variables and mathematical expressions in the input file, complicated crystal structures can be set up easily. These applications are described here.

Running a Jobs From a Shell Script

As described when discussing the syntax, a calculation can be controlled from the operating system by using environment variables. One important application is to perform a series of related calculations. Basically, a loop inside a shell script varies some parameter over the desired set of values. The values are communicated to the program using environment variables, and the programs are executed for each set. For the silicon input file discussed in Section 4.3.2, the C-shell script would look something like this, using the `n` command (shell script) from Section 4.2.2:

```
#!/bin/csh -f
# calculate E(alat) for Si
   setenv LMJOB si
   foreach alat ( 10.0 10.1 10.2 10.3 )
      setenv ALAT $alat
      n fa
      n fp
   end
```

As a further example, say we want to perform a systematic set of calculations for the dependence of the energy on the augmentation parameters `rsma` and `kmax`. Near the top, the input file contains these lines:

```
  set RSMA = 1.25
  set KMAX = 4
  getenv RSMA,KMAX
```

The first two lines are more of a fancy embellishment. First, they are a reminder of suitable values. Secondly, they can be activated by commenting out the `getenv` line (by putting a `%` in front of it) thereby making the file independent of the environment again. Thirdly, the environment variables might not be set, *e.g.*, when running the programs from the command line instead of from a shell script. In such a case the programs will give a warning and continue, using the values from the `set` statements.

In normal operation, RSMA and KMAX are read from environment variables with the same name. To make sure that the result of each job can be identified later on, the id line can be set as follows:

```
   id  < Si rsma="RSMA" kmax="KMAX">
```

The purpose is that the variable settings are displayed in the <id>.log and <id>.sum files. The augmentation parameters are defined with the line

```
  aug   {rsma} RSMA  {kmax} KMAX  {lmax} 2 {pnu} 3.5 3.5 3.3
```

Hereby the values retrieved from the environment are substituted in the input line and read by the program. Of course, we could also use some mathematical expression instead of the environment variables themselves. The corresponding shell script for the job might look as follows, typically placed in an executable file with a name such as sijob:

```
#!/bin/csh -f
#  si augmentation convergence test
   setenv LMJOB si
   n qp
   n fa
   foreach rsma ( 1.0 1.2 1.4 )
      setenv RSMA $rsma
      foreach kmax ( 2 4 6 8 )
         setenv KMAX $kmax
         n fp
      end
      echo ' ' >> si.sum
   end
```

By typing "sijob" on the command line, this script is executed. It first runs the programs for the k points and the free atoms, nqp and nfa. Since the environment variables RSMA and KMAX are not yet defined at this point, the programs will complain but nevertheless continue, using the defaults in the input file. This is not a problem because these two program do not care about these parameters. Next, a double loop is executed, changing the two parameters as prescribed and passing them to the input file via the environment variables. For clarity, an empty line is written to the summary file at the end of each loop over rsma.

A note concerning UNIX and the C shell: it is important to distinguish between shell variables (such as rsma) and environment variables (such as RSMA). Shell variables are only defined locally within the shell, but can be used in simple mathematical operations and in constructs such as foreach.

Environment variables are used to pass information between processes, since they are inherited by a child process. The value of a variable is accessed by putting `$` in front of the name. A shell variable can be set explicitly in the form "`set rsma = 1.25`" while an environment variable is set using `setenv` without an equal sign. To reduce the confusion, a standard convention is that all environment variables have uppercase names. The operating system uses environment variables such as `TERM`, `HOME`, `PATH`, `SHELL`, and `MAIL` for its own purposes; these cannot be used to pass data to a `mas` file.

Combining Job Commands and Data in One File

It is a pleasant feature if a job can be completely described by a single input file. For this purpose, a further enhancement lets the lines for the job shell script be embedded in the `mas` file. Using a single command, these lines are extracted into a separate file and executed.

As an example, suppose we want to calculate the equilibrium lattice constant of (as usual) silicon. For variety, the lattice constant is here defined by the volume relative to the experimental equilibrium. The relevant part of the input file is as follows:

```
%  Silicon energy(volume) curve

% --- job lines
   job
      n fa
      n qp
      foreach x ( 0.85 0.9 0.95 1.00 1.05 )
         setenv VOL $x
         n fp
      end
      echo 'done' >> $id.sum
      exit
   endjob

% --- data lines
   set VOL = 1.0
   getenv VOL
   set FAC = VOL^(1.0/3.0)
   set ALAT = fix4 (FAC * 10.2)

   id       < Si v/v0="VOL" alat="ALAT" >
   lattice   {a} ALAT   {vecs} 0 .5 .5   .5 .0 .5   .5 .5 .0
     ...
```

To run the whole job sequence, set `LMJOB` to `si` and type "`n exec`" on the command line. This will first extract the lines between the `job` and `endjob` statements and copy them verbatim into file `si.job`. A small awk program `job.awk` does this. At the top of `si.job`, it adds a few lines which set the environment variable `LMJOB` and the shell variable `id` to the string `si`. Secondly, the resulting script `si.job` is executed. The programs themselves skip over the `job` ... `endjob` block when reading the input file, interpreting all other lines as has been described previously. For the user, the whole procedure is equivalent to the simple functionality "execute the commands in `si.mas` for the data in `si.mas`."

Some further notes concerning this example:

- To extract the job shell script `si.job` without executing it, the command "`n -n exec`" can be used.
- Lines in the job block which refer to the input or output files could just as well be written without using the `$id` variable, for example

  ```
  echo 'done' >> si.sum
  ```

  in place of the corresponding line above. However, if file `si.mas` is then used as a model for some other calculation, it will almost certainly be forgotten to change "`si.sum`" to the new file name. By using `$id` throughout, a job block can be copied to another `mas` file without the danger of inadvertently changing any files of an older job.
- To set the lattice constant, the line

  ```
  set ALAT = fix4 (FAC * 10.2)
  ```

  was used. The `fix4` function truncates the argument to 4 digits behind the decimal point. This is more of an aesthetic question, but it avoids values such as `9.056620818`, using `9.0566` instead.

Setting Up Complicated Geometries

As described in Section 4.3.2, variables can be defined and mathematical expressions can be evaluated within the input file. A typical application is to set up complicated geometries in terms of a few parameters which characterize it. These parameters are either assigned values explicitly in the input file or are passed through as environment variables.

For example, consider the case of alpha quartz. The unit cell of this structure is hexagonal and contains three SiO_2 units. The atomic positions are specified by four parameters: u, x, y, and z. Hereby u fixes the three Si atoms and x, y, z determine the positions of the O atoms. These coordinates are defined relative to the basis vectors of the unit cell and must be transformed to cartesian coordinates at some point. The relevant lines of the input file are as follows:

```
%  SiO2 alpha quartz

   set ALAT=9.2899, CBYA=1.0996
   set U=0.4697, X=0.4135, Y=0.2669, Z=0.1191

   id   < SiO2 alfa a="ALAT" c/a="CBYA" >
   lattice  {a} ALAT  {vecs}  1 0 0  -0.5 sqrt(3)/2 0   0 0 CBYA
   symgroup  < r3z:t(0,0,"CBYA/3")  r2x >
    ...
   set T=1.0/3.0

   xpos  <Si>   -U   -U   T
   xpos  <Si>    U    0   0
   xpos  <Si>    0    U   2*T

   xpos  <O>     X      Y     Z
   xpos  <O>     Y-X   -X     Z+T
   xpos  <O>    -Y      X-Y   Z+2*T
   xpos  <O>     X-Y   -Y    -Z
   xpos  <O>     Y      X     2*T-Z
   xpos  <O>    -X      Y-X   T-Z
    ...
```

At the top of the file, the lattice constant a, the c/a ratio, and the position parameters are defined. As usual, the `id` line is used to display selected parameters in the `log` and `sum` files. The `lattice line` defines the hexagonal unit cell in terms of `ALAT` and `CBYA`. A valid symmetry generator is the three-fold rotation around (001) followed by a vertical shift of $c/3$. The `symgroup` line defines this operation correctly for any value of `CBYA`. Finally, the atoms are positioned. By using `xpos` statements and mathematical operations, the expressions for the coordinates can be entered directly. For brevity, the number one-third[3] was abbreviated by `T`.

[3]A reminder: the assignement "`set T = 1/3`" would not lead to the desired result, since integer arithmetic would assign `0` to `T`.

The crystal structure is defined completely by the six parameters `ALAT`, `CBYA`, `U`, `X`, `Y`, and `Z` appearing in the first two `set` statements. The important point is that all derived quantities such as `CBYA/3` are not entered directly as numbers. Instead, they are calculated inside the file. In this way, consistent input data is generated for any values of the six structural parameters. To vary the structure, it is enough to change only the corresponding variables, either explicitly or by using a `getenv` statement.

4.5.3 Tuning for Fast Execution

Tuning the input parameters for fast program execution is a matter of judicious tradeoff between accuracy and speed. If all convergence parameters are chosen generously, the converged answer will always be obtained, but the effort is probably considerably higher than necessary. A disadvantage of more complicated methods such as LMTO is that there are numerous parameters which influence the tradeoff. In contrast, a plane-wave pseudopotential calculation is converged by simply increasing the plane-wave cutoff until convergence is reached. This assumes, of course, that a suitable pseudopotential has been constructed; some convergence issues lurk there.

Fast execution is usually of top priority only for large systems. For these, only the steps inside the k-point loop are relevant. This is true even when a very small number of k points is used, since all steps scaling as the third power of the system size are here. More exactly, these steps scale as the third power of the matrix dimension, so the first and foremost optimisation is to try to reduce the basis size (see Section 4.4.5). The issue under discussion here is to adjust the numerical parameters for a further gain in speed. The procedure is to first determine which routines use up most of the cpu time, then to make these steps faster.

To obtain information about the time spent in the different steps, the trace profile should be switched on. This prints a summary at the end of the `nfp` run, showing the time spent in important routines. This information is also written to the `<id>.tmp` file. As a basis for the following description, such a profile is shown in Figure 4.5. The first step in the optimisation is to set up a resonably large unit cell and to generate such a profile.

Inside the k-point loop, three main tasks are done. These are: to set up the hamiltonian and overlap matrices (routine `mk_hamiltonian`), to diagonalize the matrix problem (`diagcv`), and to add to the output charge density and the forces (`add_density`). In the present example, each of these routines is called only once, because a single k-point took care of the k-space integration. The times spent in the three routines add up to approximately 80% of the total execution time. For larger systems, this percentage will

```
subroutine calls to depth  2
  wksp in  wksp out   calls   tm/call  tm total
       .0    6615.4       1   1207.31   1207.31  main
    403.2    1153.4       1     96.14     96.14    rdovfa
   1735.0    2110.5       2     83.29    166.58    mk_potential
   1741.4    1907.5       2     48.33     96.67      ves_smooth
   1907.5    2110.5       2       .44       .88      sxc_smooth
   2110.5    2110.5       2     34.08     68.15      locpot
   3119.5    6615.4       1    374.72    374.72    mk_hamiltonian
   3119.5    3139.1       1     42.42     42.42      smhs_q
   3139.1    6615.4       1    250.26    250.26      hsi_q
   6615.4    6615.4       1     81.80     81.80      augm_q
   6615.4    6615.4       1    277.34    277.34    diagcv
   6615.4    6615.4       1    279.94    279.94    add_density
   6615.4    6615.4       1     55.18     55.18      fsm_q
   6615.4    6615.4       1    152.43    152.43      rsif_q
   6615.4    6615.4       1     72.33     72.33      rloc_q
   6615.4    6615.4       1      8.12      8.12    mk_density
   6615.4    6615.4       1       .20       .20    sym_rhoat
   6615.4    6615.4       1       .55       .55    sym_smrho
```

Figure 4.5: Trace profile showing the time spent inside the various program steps. Routines called by a higher-level routine are shifted to the right. The columns contain the following data: the high-water mark for workspace at first entry and exit of the routine, the number of times the routine was called, the average time per call, and the total time spent in the routine. Workspace is given in units of single-precision reals and execution time is in seconds.

become even higher.

Possibly optimisation of the diagonalisation step `diagcv` can be done by switching to a faster matrix diagonalisation routine, for example a special routine available on a specific computer. This will not be discussed here. For both the hamiltonian setup and the output density evaluation, three separate substeps are needed. For the case of `mk_hamiltonian`, these are routines `smhs_q`, `hsi_q`, and `augm_q`. The corresponding steps in `add_density` are `fsm_q`, `rsif_q`, and `rloc_q`. The optimisation of these steps is discussed separately below.

Steps `smhs_q` and `fsm_q` – These routines calculate terms which can be done analytically for the smooth Hankel envelope functions, namely the overlap and kinetic energy matrices and the associated force terms. For each pair of atoms in the unit cell, Ewald summation is used to evaluate a Bloch function of the connecting vector between the two sites. The effort for a single Ewald sum is approximately independent of the unit-cell size, so the overall effort in these steps only scales as N^2. For sizeable systems, the time spent here is therefore not very significant. Should it be desirable to optimise these steps for some reason, the only way is to speed up the Ewald summation. By increasing the Ewald tolerance, the number of real-space and recirocal-space vectors is reduced. There might be some gain in changing the partition into real-space and reciprocal-space sums.

Steps `hsi_q` and `rsif_q` – These routines evaluate quantities over the real-space mesh. During the matrix setup, `hsi_q` numerically integrates the matrix elements of the smooth potential (Eq. 3.77). For the output density, `rsif_q` makes smooth wavefunctions by summing the smooth Hankel envelopes, then adds the moduli squared to the smooth output mesh density. Associated force terms are also calculated. Generally a large portion of the total execution time is spent in these routines. Obviously, the amount of time used depends mainly on the fineness of the real-space mesh. Basically, optimisation is done by making the mesh coarser until the error starts to become larger than acceptable.

However, this is not the whole story. The required fineness of the mesh is determined by the smallest smoothing radius used in the basis. The time-intensive steps are actually done in reciprocal space, involving the Fourier transforms of the smooth Hankel envelopes (Eq. 3.78). These decay as gaussians with a rate determined by the smoothing radius. Thus, it is adequate to include only the reciprocal vectors within a smaller cutoff for those functions which have a larger smoothing radius. By exploiting this, an increase in execution speed can be obtained, especially when the basis contains only a few functions with tight smoothing. This might be the case if small and large atoms (say, silicon and hydrogen) are in the system at the same time. However, a gain is also evident when different smoothing radii are used on the same atom.

In practice, orbital-dependent cutoff radii are calculated such that the relevant Fourier transform is smaller than a given tolerance outside the cutoff sphere. This tolerance can be specified as an optional parameter in the `ftmesh` input line. For example, for silicon with a real-space mesh of $12 \times 12 \times 12$ points, the program uses 609 recip vectors within a cutoff of 5.25 to represent the Fourier transform of a function. A suitable basis might contain functions of smoothing radii 1.40 and 2.20. The orbital-dependent cutoffs are shown in the listing:

```
su_gvcut: make orbital-dependent recip cutoffs for tol= 1.00D-07
spec  blk    eh     rsm   l1 l2     gmax     last term    cutoff
 si    1   -.50   1.40   0  2     6.360    8.33D-05     609*
 si    2  -1.40   2.20   0  1     3.798    2.45D-07     259
```

For the smoothing radius 1.40, efficiency is not increased because all 609 recips already lie inside the calculated cutoff of 6.360. This is indicated by the star. For the larger smoothing radius of 2.20, the orbital-dependent cutoff sphere only contains 259 vectors. Thus, operations for these basis functions are done more efficiently. If the tolerance is made larger, the cutoff spheres become smaller and more time can be saved. As usual, the idea is to find a suitable compromise between the numerical effort and the errors due to the neglected terms.

In sum, the orbital-dependent cutoffs place a ceiling on certain time-consuming steps within the k-point loop. If the real-space mesh is made finer, these steps will not become slower once these cutoffs are reached. Of course, other mesh-dependent operations which did not play a prominent role before (such as the fast Fourier transforms) will slow down and start to consume an increasing portion of the cpu time. If the trace profile is requested to a deeper level, this can be inspected in detail.

Steps `augm_q` and `rloc_q` – These routines are concerned with the augmentation, whereby `augm_q` adds the augmentation terms to the overlap and hamiltonian matrices and `rloc_q` makes the local atomic densities and the associated force terms. Depending on the input parameters, these steps might consume a significant portion of the cpu time. The most important parameter are the cutoffs `lmxa` and `kmax` for the augmentation expansion. Thus it is worth the effort to determine the lowest values which give adequate accuracy. Hereby the angular-momentum cutoff `lmxa` plays the more important role. Secondly, augmentation is done using local expansions around the sites which are obtained using Ewald summation. Thus, as described for `smhs_q` and `fsm_q`, some cpu time can be saved by increasing the tolerance for the Ewald summation.

Chapter 5

Special Topics

This chapter discussed some important topics in more detail. These are: the Brillouin zone integration, the calculation of the forces, and the frozen overlapped core approximation. The last two topics concern features new to this program. Some open questions remain here, which will hopefully be answered as experience for different systems accumulates.

5.1 Brillouin-Zone Integration

According to the density-functional equations, we should calculate and occupy eigenstates until all electrons have been accomodated, starting at low energies and working up. For a periodic system such as a crystal, this discrete sum over the eigenstates turns into an integral over the Brillouin zone together with a summation over the various bands. States are now occupied up to the Fermi energy, by definition the energy for which the total valence charge has been accomodated.

In practice, this is not an easy affair. For a given Fermi energy, the states in the Brillouin zone can be classified by those whose eigenvalue which lie above it and those which lie below. The boundary between these states in k space is the Fermi surface. To calculate any integral over the occupied states, we must integrate over the volume enclosed by it. This is a three-dimensional integral over a region within a complicated boundary, well-known to be a difficult problem for numerical techniques.

In any case, we start by placing a regular mesh through the Brillouin zone and calculate the eigenstates for these discrete k vectors. The further procedure then depends to a large extent on whether the system is a semiconductor or a metal. Generally most of the computer time is spent in the main loop over the k points, and it is important to keep the number of points required minimal.

5.1.1 Semiconductors

For a semiconductor, pleasantly, the problem disappears by itself. The Fermi energy lies in the gap between two bands. Any given band is either completely full or completely empty. Thus, the Fermi surface has shrunk to zero and the integrals go over the whole Brillouin zone. Numerically, this is a completely different situation. To integrate a smooth periodic function over the full periodicity volume, the best approach is to simply sum over the points of a regular mesh, assigning all points the same weight. In fact, this procedure converges faster than any power of the mesh spacing (assuming the integrand is analytic). There is no advantage whatsoever in using more complicated methods such as three-dimensional Simpson integration[1] The fast convergence with the mesh fineness leads to accurate results for a ridiculously small number of mesh points. Typically, two to four divisions along each edge of the reciprocal unit cell are already enough.

This result leads directly to the important "special points" meshes. As described further on, it is redundant to recalculate the eigenstates for a k vector which is related to a previous, already calculated one by a symmetry operation of the crystal. Instead, only one representative is calculated and the others are taken into account by increasing the weight. For a given mesh fineness, it saves computation time by choosing the mesh so that each point is related to as many others as possible by the symmetry. For example, it is clearly a bad idea to include $\Gamma = (0, 0, 0)$ since this point is in a class by itself, as compared to others which belong to a class of up to 48 equivalent points.

We always want to integrate using a regular mesh, but there is still the freedom to shift a given mesh by an arbitrary vector. Furthermore, the used mesh does not have to be a "miniature" version of the reciprocal lattice; cubic meshes can be used for fcc and bcc as well as sc. Using this freedom, optimal special points meshes have been defined for the different lattices. The meshes can be constructed by cutting each edge of the reciprocal unit cell into an even number of divisions, then leaving away selected points depending on the lattice type. In each case Γ is not included. For example, for fcc a reasonable underlying mesh might have four divisions along each edge. Assuming the full 48-element cubic symmetry group, the final special-point mesh consists of only two inequivalent points, namely $(1/4, 1/4, 1/4)$ and $(1/4, 1/4, 3/4)$ with weights of 1/4 and 3/4, respectively. These two points

[1]Consider a periodic one-dimensional integral. Simpson integration has weights 2/3 and 4/3 for alternating points. There are two equivalent ways to do this, with 1/3 on the even respectively the odd points. A better way than either is to average over the two choices, which assigns the same weight to all points. This can also be done for a non-periodic situation if the integrand is smooth and has gone to zero at the limits.

carry all the information for a regular mesh of 32 points in the Brillouin zone.

In summary, the Brillouin zone integration for a semiconductor is easy to do. Since all integrals run over a full band, there is no Fermi surface and it is sufficient to use a regular mesh with a small number of points. By shifting this mesh off Γ and exploiting the symmetry, only very few non-equivalent k points actually have to be considered. For each of these points, the eigenstates of the lowest $Q_{\text{val}}/2$ eigenstates are added into the charge density and the eigenvalue sum.

5.1.2 Metals

For two reasons, the Brillouin zone integration is significantly more difficult for metals than for semiconductors. First of all, the k points cannot be handled independently of each other. Information from all the points is needed in order to determine the Fermi energy, and only then can the information from the separate points be added into the eigenvalue sum and charge density. Secondly, as discussed, integrals must be evaluated over the irregularily-shaped part of reciprocal space inside the Fermi surface.

Homing in on the Fermi energy

The first problem can be solved in principle as follows. Since we do not know the Fermi energy ahead of time, we tabulate the relevant quantities as a function of the energy. After the Fermi energy has been determined, we look up the desired integrals in the tables. For example, we can accumulate the density of states $D(\epsilon)$ on an energy mesh of, say, one-thousand points. After the DOS has been built up using information from all the k points, the Fermi energy is fixed by

$$Q_{\text{val}} = \int_{-\infty}^{\epsilon_{\text{F}}} D(\epsilon)\, d\epsilon \ . \tag{5.1}$$

One of the quantities we need, the sum of the occupied eigenvalues, is then given by

$$E_{\text{val}} = \int_{-\infty}^{\epsilon_{\text{F}}} D(\epsilon)\epsilon\, d\epsilon \ . \tag{5.2}$$

The practical problem is how to handle the charge density in this vein. We could use the identical procedure, but this would involve an expensive energy-resolved tabulation for each of the many coefficients needed to specify the density.

To streamline this, the energy-resolved charge density is tabulated on a minimal mesh of only three points. These are the energies $\epsilon_1 = \epsilon_{\text{F}}^0 - \delta\epsilon_{\text{F}}$,

$\epsilon_2 = \epsilon_F^0$, and $\epsilon_3 = \epsilon_F^0 + \delta\epsilon_F$, where the two numbers ϵ_F^0 and $\delta\epsilon_F$ are initially specified in the input file. For each energy ϵ_i we calculate the charge density $\rho_i(\mathbf{r})$ which would result if it was known beforehand that this was the correct Fermi energy. Once the final Fermi energy is known, we interpolate to that energy from the three-point energy mesh. If Q_1, Q_2, and Q_3 are the charges associated with the three guessed Fermi energies, a quadratic interpolation $Q(\epsilon)$ is laid through the three points. The output Fermi energy is estimated by the solution of the quadratic equation $Q(\epsilon) = Q_{\text{val}}$. This defines three weights w_1, w_2, and w_3 with the properties

$$w_1 + w_2 + w_3 = 1$$
$$w_1 Q_1 + w_2 Q_2 + w_3 Q_3 = Q_{\text{val}} \ .$$

The output charge density is finally assembled according to

$$\rho(\mathbf{r}) = w_1 \rho_1(\mathbf{r}) + w_2 \rho_2(\mathbf{r}) + w_3 \rho_3(\mathbf{r}) \ . \tag{5.3}$$

The eigenvalue sum is interpolated in a similar manner.

To ensure that the interpolation from the three-point energy mesh does not introduce relevant errors, the window is adjusted to home in on the final Fermi energy. After each iteration, the central energy ϵ_F^0 is set equal to the newly determined Fermi energy. The width $\delta\epsilon_F$ is made smaller in proportion to how much the Fermi energy has moved. Near convergence, this gives a very narrow window closely around the (now hopefully stable) Fermi energy. The sign that this is indeed happening is that weight w_2 approaches 1.0 while w_1 and w_3 are close to zero.

This procedure is robust in practice, but some points must be kept in mind:

- In the first iteration, the input values of ϵ_F^0 and $\delta\epsilon_F$ should define a relatively wide window which brackets the output Fermi energy with reasonable confidence. If it is completely unknown where this Fermi energy will land, it is best to do a preliminary iteration to get this information.

- While iterating, check that the three-point energy window behaves as expected. The interpolated Fermi energy should lie inside the window, the window should become very narrow, and the weight for the middle energy should approach 1.0.

- It is possible to change various parameters (for example, the basis size) in the middle of a calculation. This is done by writing the restart file, changing the parameters in the main input file, and continuing from there. Since the Fermi energy will generally jump by some amount, it is important that the window width $\delta\epsilon_{\mathrm{F}}$ is large enough. If the window has become very narrow and the Fermi energy shifts significantly, the well-behaved interpolation procedure turns into a wild extrapolation. Similar precautions are needed if a self-consistent calculation is used as starting point for a different geometry.

Gaussian smearing

It has not been answered yet how to make the density of states and the energy-resolved charge density in the first place. This is the second question mentioned above: how to evaluate an integral which goes over only that part of k-space inside the Fermi surface. In the present case, these integrals are needed for the total charge, the eigenvalue sum, and the charge density associated with the three guessed Fermi energies.

One perfectly good way to proceed is to use the linear tetrahedron method [19]. The Brillouin zone is cut up into tetrahedra such that all corners are mesh points. Inside each tetrahedron, the integrand is interpolated linearily between the corner values. This representation is integrated analytically, one tethrahedron at a time. Good accuracy is achieved in this way, especially using a cleverly modified variant [20]. As disadvantages, quite a lot of bookkeeping is needed, problems arise if there are band crossings near the Fermi energy, and exact agreement between the forces and the total energy is difficult.

The alternative used here is the "smearing" procedure. Instead of a sharp cutoff at the Fermi energy, the occupation is changed smoothly from one to zero in a range around the Fermi energy. The integrand in k-space has now been turned into a smooth function and, as in the case of semiconductors, the integral converges quickly with increasing mesh fineness. The smearing can be done using the standard finite-temperature Fermi-Dirac distribution, but other smooth approximations to the step function work equally well. For gaussian smearing, the smoothed-out step function is defined using the error function:

$$f(\epsilon) = \tfrac{1}{2}\left[1 - \mathrm{erf}\,(\epsilon/\sigma)\right] \tag{5.4}$$

where σ is the energy range in which the function goes from one to zero.

It is trivial to implement the smearing procedure. Each eigenstate is assigned a weight $f(\epsilon_i(\mathbf{q}) - \epsilon_{\mathrm{F}})$ depending on the position of the eigenvalue $\epsilon_i(\mathbf{q})$ relative to the Fermi energy. After that, the integrals are made by simple summation:

$$Q_{\mathrm{val}} \;=\; \sum_{\mathbf{q}} \sum_i w(\mathbf{q}) f(\epsilon_i(\mathbf{q}) - \epsilon_{\mathrm{F}}) \tag{5.5}$$

$$E_{\mathrm{val}} = \sum_{\mathbf{q}} \sum_i w(\mathbf{q}) f(\epsilon_i(\mathbf{q}) - \epsilon_{\mathrm{F}})\, \epsilon_i(\mathbf{q}) \tag{5.6}$$

$$\rho(\mathbf{r}) = \sum_{\mathbf{q}} \sum_i w(\mathbf{q}) f(\epsilon_i(\mathbf{q}) - \epsilon_{\mathrm{F}})\, |\psi_{\mathbf{q}i}(\mathbf{r})|^2 \tag{5.7}$$

where $\mathbf{q}$ runs over the irreducible k-space mesh, the $w(\mathbf{q})$ are the corresponding weights derived from the lattice symmetry, and i runs over the energy bands.

The density of states is the derivative of Q_{val} respective to the Fermi energy, *i.e.*,

$$D(\epsilon) = \frac{1}{\sigma\sqrt{\pi}} \sum_{\mathbf{q}} \sum_i w(\mathbf{q}) \exp\left[-(\epsilon - \epsilon_i(\mathbf{q}))^2/\sigma^2\right] \,. \tag{5.8}$$

This shows that the DOS is built up as a kind of histogram: for each eigenstate, a gaussian centered at the eigenvalue and of width σ is added into $D(\epsilon)$. It is easily appreciated that for smaller smearing widths σ, more and more k points are needed to get a reasonable representation.

In the smearing method, a very simple modification transforms the previously undoable integral into one which converges exponentially with the mesh fineness. The catch is that the result (although converged as a Brillouin-zone sum) is not the exact ground-state energy. Often it is a problem to find a smearing width which allows acceptable k-space convergence but does not cause unacceptable errors in the total energy. Methods have been developed to get around this, as will be described next.

Entropy and Higher-Order Smearing

When smearing is used, it has been shown that the energy which is variational [2] is actually the electronic free energy:

$$F = E - \sigma S(\sigma) \tag{5.9}$$

where E is the previously-considered total energy and $S(\sigma)$ is a suitable entropy term. For Fermi-Dirac broadening, the entropy is given by the standard expressions of thermodynamics, and for gaussian smearing, the entropy associated with an eigenvalue ϵ_n is

$$S = \frac{1}{2\sqrt{\pi}} \exp\left(-\left(\frac{\epsilon_n - \epsilon_{\mathrm{F}}}{\sigma}\right)^2\right) \tag{5.10}$$

[2] Making a system selfconsistent is equivalent to minimising some energy expression, the variational energy. A proper minimisation is needed for fast convergence and accurate forces.

Depending on one's background, it may or may not be surprising to find the entropy suddenly turning up in this context. Since a clean presentation has been done elsewhere [21], we can try to understand what is going on in more elementary terms.

At present, there are two different candidates to calculate the sum of the eigenvalues up to the Fermi energy. First, we can sum over the eigenvalues according to (5.6). Second, we can accumulate the density of states according to (5.8), then calculate the integral $\int^{\epsilon_{\mathrm{F}}} D(\epsilon)\epsilon \, d\epsilon$. The interesting point is that these two numbers are different, but that by adding the entropy term, the first prescription produces the same result as the second.

More exactly, the contributions to the eigenvalue sum from an eigenvalue ϵ_n using the first and second prescription are, respectively,

$$\epsilon_n \int^{\epsilon_{\mathrm{F}}} g(\epsilon - \epsilon_n) \, d\epsilon \tag{5.11}$$

and

$$\int^{\epsilon_{\mathrm{F}}} \epsilon \, g(\epsilon - \epsilon_n) \, d\epsilon \ . \tag{5.12}$$

where $g(\epsilon - \epsilon_n)$ is the contribution to the DOS as in (5.8). In the first case, the full weight associated with this eigenstate is added in at the eigenvalue ϵ_n. In the second case, the integral over the energy times the DOS contribution is done properly. The difference of the two terms is

$$\int^{\epsilon_{\mathrm{F}}} (\epsilon - \epsilon_n) g(\epsilon - \epsilon_n) \ . \tag{5.13}$$

This is zero when ϵ_n lies far below the Fermi energy and the state is fully occupied, but not if ϵ_n is in the vincinity of the Fermi energy. Using the definition of the entropy, it is easy to see that $-\sigma S$ is exactly this integral.

In summary, the "purpose" of the added entropy can be formulated like this. For non-zero smearing, the calculated DOS is some more or less smoothened version of the correct one. By including the entropy term, we ensure that this approximate DOS is being filled up to a *sharp cutoff* at the Fermi energy. Furthermore, this sharp cutoff is important for obtaining correct forces. For a pertubation of the external potential (such as the shift of an atom), the first-order change of the self-consistent total energy is

$$\delta E = \sum \epsilon_n \delta q_n + \int \rho(\mathbf{r}) \delta V_{\mathrm{ext}}(\mathbf{r}) \, d\mathbf{r} \tag{5.14}$$

where the δq_n are the changes of the occupation numbers due to the pertubation. A valid force theorem evaluates the integral $\int \rho \delta V_{\mathrm{ext}}$ properly, but there is no easy way to include the occupation changes as the bands distort in response to the pertubation. Luckily, for a sharp Fermi cutoff the changing occupations all enter at the Fermi energy. The sum reduces to $\epsilon_\mathrm{F} \sum \delta q_n$ which is zero because the total charge is conserved.

Finally, in order to reduce the remaining errors due to the somewhat wrong shape of the DOS, two approaches can be used:

1. First, it has been shown [22] that to quadratic order in σ, the total energy and the free energy behave as

$$\begin{aligned} E(\sigma) &= E_0 + \tfrac{1}{2}\sigma S(\sigma) &\quad (5.15)\\ F(\sigma) &= E_0 - \tfrac{1}{2}\sigma S(\sigma) &\quad (5.16) \end{aligned}$$

   where E_0 is the energy at zero broadening. Since both E and F are available in the calculation, a good estimate of the ground-state energy can be obtained as

$$E_0 \approx \tfrac{1}{2}[E(\sigma) + F(\sigma)] \; . \tag{5.17}$$

   This method gives accurate values for the zero-smearing energy even for large smearing widths. However, it is a problem that the forces, being the derivatives of F, do not exactly match to the calculated ground-state energy, E_0.

2. Second, several problems can be avoided simultaneously by using special higher-order gaussian smearing functions [23]. A suitable entropy term can be defined for these also. It turns out that the entropy term is very small[3] for a reasonable choice of the parameters[21]. Although the extrapolation to zero broadening could be done here also, this step is generally not necessary since $E(\sigma)$, $F(\sigma)$, and E_0 are all close together anyway. Under these circumstances, the calculation produces the correct ground-state energy together with the associated forces. Thus, this approach seems preferable whenever the forces are important.

Both of these approaches only work if the DOS is smooth in the vicinity of the Fermi energy, on the scale of the smearing width. If the Fermi energy happens to lie at or very close to a Van Hove singularity, be prepared to use a considerable number of k points in any approach.

[3]The entropy for the Nth order is proportional to $H_{2N}(x)\exp(-x^2)$ where $x = (\epsilon - \epsilon_\mathrm{F})/\sigma$ (see Ref. [21]). This is orthogonal to polynomials of order lower than $2N$. Thus the entropy is zero if the DOS in the smearing range can be described by such a polynomial.

Rules of Thumb

This discussion of the Brillouin-zone integration in metals has been somewhat lengthy. To come back down to the ground, here are some rules of thumb to help choose the parameters.

- Even using the smearing technique, a metal generally requires considerably more k points than a semiconductor. For a system with one atom per unit cell, results *begin* to make sense at perhaps 8–12 divisions along the edge of the reciprocal cell, using the normal mesh containing Γ. The resulting mesh has 512–1728 points which are reduced to about 30–70 irreducible points, assuming full 48-element cubic symmetry. Depending on the application, considerably more k points might be needed, possibly up to 30 divisions along each edge.

- Since a relatively fine mesh is being used, there is often not much gain in using a special-points mesh for unit cells containing only a few atoms.

- The width of the energy smearing σ depends on the type of band structure. For wide *sp* band as in Al, typically 40 to 80 mRy can be used. For narrower bands as in the transition metals, the smearing is typically in the range of 15 to 25 mRy.

- In some systems, the Fermi energy lies close to a Van Hove singularity. In that case, try to chose the smearing width smaller than the distance between ϵ_{F} and the singularity, if possible. There is not much to be done here: a large and expensive k mesh will be needed in any case.

- The order of the smearing function is not especially critical if only the total energy is of interest; it can be zero for standard gaussian broadening or 1–2 for the higher-order functions. If order zero and a large smearing is used, it is desirable to extrapolate to the zero-smearing ground-state energy as described above (by whipping out a pocket calculator and adding one-half of the entropy term to the calculated total energy F).

- If accurate forces are needed, a function of order one or higher should be used. The order and the smearing width should be chosen in such a way that the entropy term is small. This ensures that the calculated energy is close to the correct ground-state energy. At the same time, the forces will correspond to this energy.

5.1.3 Supercells

When calculating supercells or other large unit cells, the previous discussion is basically still applicable. In addition, some special considerations become important:

- Since the band structure is folded into a smaller Brillouin zone, each k point supplies more eigenstates and consequently fewer k points are needed. In principle, if a calculation for one atom per unit cell required Q k points, a supercell with N atoms should require Q/N points. Note that this estimate refers to the number of points in the whole zone, not in the irreducible wedge.

- Since a smaller number of k points is being used, it generally makes sense to use special points meshes in supercells, even for metals. For very large supercells, only one k point is needed. Then, it is better to use a very coarse special-point mesh instead of a mesh containing only Γ.

- In directions for which the bands have little or no dispersion, the number of k points can be very small. Specifically, for a slab it is often enough to take one plane of k points.

5.2 Forces

The derivation in Section 3.2.5 resulted in a simple force theorem. The force consists of two parts. The first contribution is a purely electrostatic term associated with the smooth mesh density, namely the force of the smooth compensated density on the gaussians which represent the core and the nucleus. The second contribution is similar to a Pulay term. The crystal potential enters the eigenvalues via the smooth mesh potential and the local augmentation matrices. The second force term gives the change in the eigenvalue sum when these quantities are frozen while the atoms are shifted. These two terms are evaluated in the program.

For a practical application, three points can become important. First, care must be taken when calculating forces for metals. Second, some other effects can make a difference to how well the forces agree with the energy. Finally, it is desirable to obtain reasonable forces even when the system is not completely selfconsistent yet.

Smearing fct.		Energy (mRy)			2nd derivative	
width	order	F(σ)	$-\sigma S(\sigma)$	E_0	E''	$-F'$
0.10	0	−656.8	−159.9	−576.8	393.0	389.7
	1	−576.6	−5.3	−574.0	423.2	419.5
	2	−575.3	1.7	−576.2	467.1	463.0
0.05	0	−596.6	−40.0	−576.6	456.7	452.6
	1	−576.4	0.7	−576.7	474.6	471.2
	2	−576.5	0.6	−576.8	459.4	456.8

Table 5.1: Calculated data for a frozen phonon in Mo as obtained using different choices of the smearing function for the Brillouin-zone integration (see text).

Forces for Metals

The connection between forces and the Brillouin-zone integration for metals was discussed in detail in Section 5.1. To summarize, the following recipe (taken from Ref. [21]) is strongly recomended. The k-space integration should be done with a smearing function of order one or higher. The criterium for properly chosen values of the smearing order and width is that the magnitude of the entropy term $-\sigma S$ is small. Under these conditions, the energy and the forces agree and are close to results for zero smearing.

To make this less abstract, Table 5.1 shows some data for the X phonon in Mo. For this phonon mode, the body-centered atom moves against the corner atoms. Each line of the table corresponds to one choice of the smearing function, for which frozen phonons were calculated at five different amplitudes. The second block of the table shows some total-energy terms for zero phonon amplitude. $F(\sigma)$ is the energy as calculated by the program, which includes the entropy term $-\sigma S(\sigma)$. The ground-state energy can be estimated by extrapolating to zero smearing using $E_0 \approx F(\sigma) + \frac{1}{2}\sigma S(\sigma)$, also shown. The third block gives the second derivative at zero amplitude (equivalent to the phonon frequency) as obtained using the energy curve and a second time using the derivative of the force.

The first point to notice is that the two values of the curvature always agree to within 1%, showing that the forces are the indeed the derivatives of the energy. This is true even when the smearing is so large that the "wrong" phonon frequency comes out. Secondly, one sees that the energy curve is substantially shifted when smearing of order zero is used, whereas the curves using higher-order smearing all lie close together. By extrapolating to E_0, an accurate energy is obtained for all smearing orders. However, this step is unnecessary for the higher-order functions. Finally, the calculated second derivative approaches a stable value to the degree that $-\sigma S(\sigma)$ is small. For

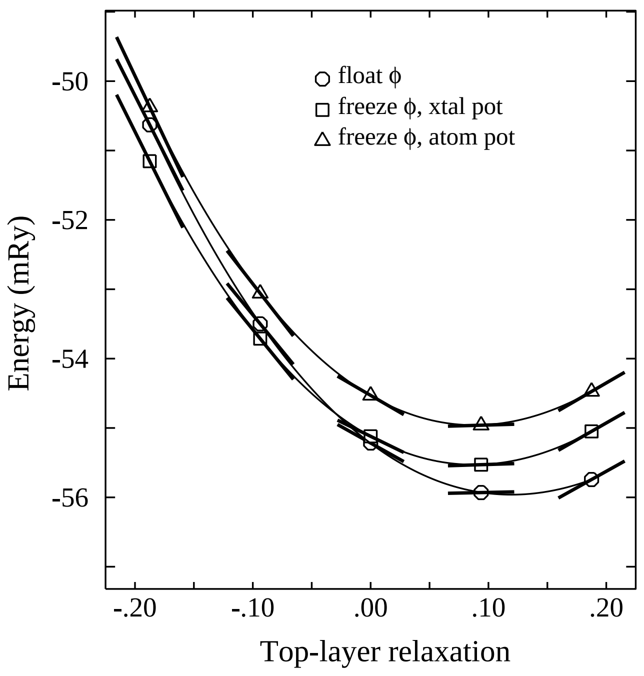

Figure 5.1: Energy and forces (tangential lines) for the top-layer relaxation of a three-layer Al slab. When ϕ_l and $\dot{\phi}_l$ are floated (circles), the forces do not quite agree with the energy. The agreement is good if ϕ_l, $\dot{\phi}_l$ are frozen after a few iterations at relaxation zero (squares). The agreement is also good when ϕ_l, $\dot{\phi}_l$ are taken from the free atom (triangles), but the energy curve is shifted slightly. This is a simple test system and the relaxation itself should not be taken seriously.

the three cases where the entropy term is smaller than 2mRy, the second derivatives agree to within 2%.

Getting the Forces and the Energy to Agree

For a metal, the first requirement for good agreement between the forces and the energy is that the Brillouin-zone integration has been converged properly. This does not mean that the smearing width must chosen in any special way; the point is that the k-space mesh must be fine enough for the smearing used in the calculation.

The second point is relevant for metals as well as semiconductors and concerns the augmentation. The force theorem takes into account the changes in the self-consistent potential as the atoms move about. When the potential is modified, the numerical solutions ϕ_l, $\dot{\phi}_l$ also change to some extent. However, the force theorem implicitly assumes that the augmentation is done by a fixed set of radial functions.

Depending on the system, it may be neccessary to keep ϕ_l and $\dot{\phi}_l$ frozen explicitly. This can be done using the `friz` switch in the `switches` line of the input file. More exactly, this switch freezes the spherical potential and the boundary conditions P_ν used when making ϕ_l and $\dot{\phi}_l$. At the same time, this freezes the core states. If this switch is used, the calculated results depend to some extent on the potential which is frozen. When this is done already in the first iteration, this potential is taken from the free atom. This is illustrated in Fig. 5.1 for a simple test system for which the effect is rather large.

Convergence of the Forces

As described above, the force consists of an electrostatic term from the smooth density plus a Pulay term. At selfconsistency, the total force is tangential to the energy surface. However, things are not as clear when selfconsistency is not yet reached. In fact, there are then two different ways to evaluate the force, depending on whether the electrostatic contribution is taken from the input or the output density. Both of these are calculated in the program. It turns out that the force calculated using the output density has a well-defined interpretation and shows better convergence with iteration number.

To illustrate this, Figure 5.2 performs in real life the *Gedanken-experiment* used to derive the force theorem in Section 3.2.5. First, germanium was made selfconsistent for a specific frozen-in optical phonon, shown near the middle of the plot. For other amplitudes of the phonon, a guessed density was made by keeping the smooth mesh density fixed and shifting the local contributions rigidly. More prosaically, this simply means that the selfconsistent restart file was kept and used as input for the other geometries. The middle curve (triangles) presents the self-consistent energy and the corresponding force. The bottom curve (circles) shows the Harris energy for the first iteration. As claimed previously, this curve is tangential to the self-consistent energy at the reference geometry. The top curve (squares) is the Kohn-Sham energy for the first iteration, which is tangential in the same way. As a note in passing, this test is a good way to check that the forces and energies are being evaluated properly.

To continue to the main point of this exercise, the forces calculated using the electrostatic term from the output density are also shown. These are the thick lines drawn on the Harris energy curve, clearly tangential to it. This demonstrates that the force expression using the output density corresponds to the Harris energy in the first iteration when the trial density is defined as described. For the force using the input density, no such interpretation

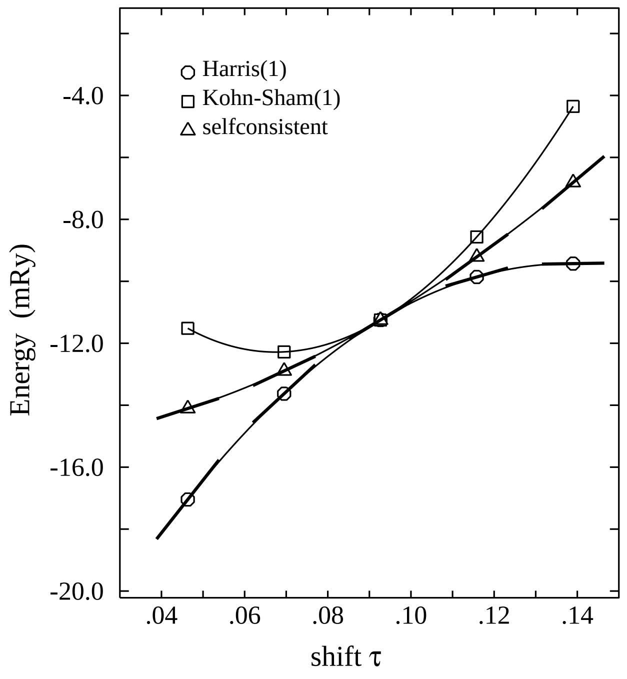

Figure 5.2: The "three-curve test" for the energies and forces in the case of the optical phonon in Ge (see text). The lines tangential to the Harris energy in the first iteration (circles) show the force calculated using the electrostatic term from the output density.

was found.

The rigidly-shifted trial density is a prescription which produces an energy curve somewhere near the selfconsistent curve, and the force derived from it is a well-defined approximation to the true one. Experience has shown that this force generally converges considerably faster than the force based on the input density. A typical example is shown in Figure 5.3. The logarithmic plot gives a more precise formulation for the improved convergence. One sees that the rate of convergence is the same for both formulations, but that the error is smaller by about one order of magnitude when the output density is used.

5.3 FOCA

For many systems, the core states are so extended that they spill to some extent out of the muffin-tin spheres. Typical problematic cases are the transition metals near the beginning of the series. Whether the core causes problems depends on the type of atom, on the size of the muffin-tin spheres,

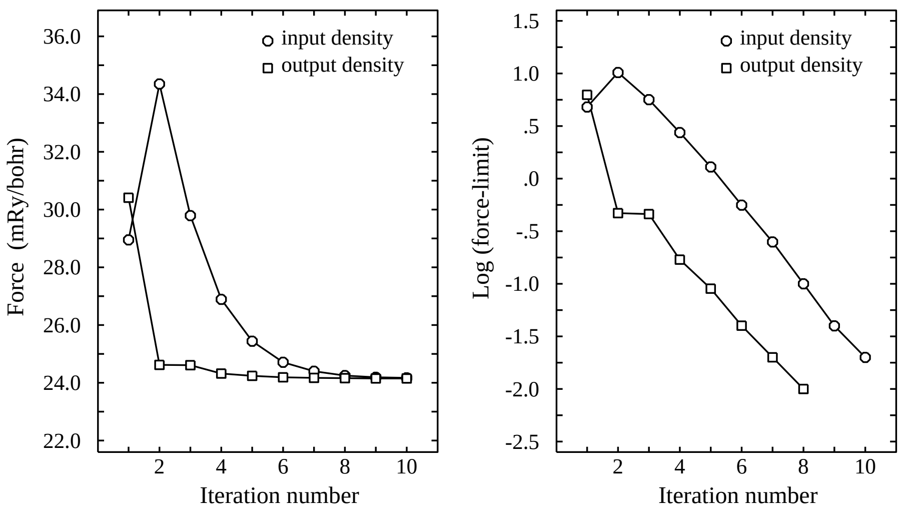

Figure 5.3: Calculated force as function of the iteration number for the Ge optical phonon when the electrostatic force contribution comes from the input density (circles) respectively the output density (squares). The right-hand plot shows the logarithms of the deviation from the converged value.

and on the choice of the valence states. The core spill-out for the free atoms and the chosen muffin-tin radius can be taken as an estimate, although the core may shrink or expand in response to charge transfer in the crystal. Some typical values are shown in Table 5.2. Experience has shown that the cores in Al, Si, and Pd and (obviously) for atoms such as O or N do not cause any problems. For Ge, Mo, and Ti, the core is already large enough for a special treatment to be warranted. In addition, there are some rather more extreme cases, such as GaN with the Ga 3*d* state in the core and Ti in Ti_2O_3 if the spheres are chosen as shown.

As long as the core states have a negligible amplitude at the muffin-tin radius, the procedure is straightforward. The core states can be calculated for an arbitrary set of boundary conditions at R_{mt} since the results are essentially independent of the choice. For example, setting the slope equal to the negative of the value is one perfectly good recipe. However, when a core state has a significant amplitude at R_{mt}, it is not clear *a priori* what to do next. The core eigenvalues and therefore the total energy now depend on the chosen boundary conditions. This dependence can be quite sensitive because of the large amount of charge involved, for example, ten electrons in the case of a *d* core. Even if reasonable boundary conditions have been found somehow, it is still not clear how to handle the charge which spills out of the sphere.

atom	R_{MT}	Q_{spill}
Al	2.50	0.002
Si	2.20	0.013
Ge	2.20	0.060
Te	2.65	0.019
Pd	2.55	0.019
Mo	2.50	0.101
Ti	2.70	0.031
Ga in GaN, $3d$ in valence	1.78	0.021
Ga in GaN, $3d$ in core	1.78	0.406
N in GaN	1.78	0.000
Ti in Ti_2O_3	2.00	0.280
O in Ti_2O_3	1.50	0.000

Table 5.2: The amount of free-atom core charge which spills out of the muffin-tin sphere for some typical systems.

Methods to Handle Semicore States

The solution implemented in this program is the "frozen overlapped core approximation," or FOCA, to be described below. Before that, here is a summary of the various methods which can be used to handle the problem of an extended semicore:

- Ignore it. If we treat the core incorrectly but at least consistently, total-energy differences might be acceptable. For example, we could prescribe the the value of the core wavefunction at R_{mt} to be zero. In a slightly better procedure, the program uses the following procedure when the FOCA is switched off. The core boundary conditions are determined by attaching a tail function which is the correct solution for a constant potential outside the sphere, equal to the potential at R_{mt}. This function is yet again (aarrgh!) a Hankel function. The prescription leads to core eigenvalues which are relatively robust when the muffin-tin sphere starts to squeeze on the core. They can be inspected by setting the verbosity large enough to print the core eigenvalues, in a table such as

```
state  chg        ecor0        ecore        tcore   nre  rho(rmax)
1s    2.00  -353.769285  -353.769285   461.569821   205   .00000
2s    2.00   -38.236519   -38.236519    79.801172   239   .00000
3s    2.00    -3.947085    -3.940797    16.363755   239   .00013
2p    6.00   -31.881335   -31.881335    78.554428   239   .00000
3p    6.00    -2.291389    -2.283372    13.975975   239   .00129
```

Here `ecor0` is the eigenvalue when the wavefunction at R_{mt} is zero, and `ecore` corresponds to the described Hankel boundary conditions. To make the output density, the full core charge is placed inside the sphere. Needless to say, the total-energy differences will be most reliable if the muffin-tin radii for a certain type of atom are kept fixed.

- Solve the problem properly. As described by Singh [24], the correct solution is to include functions of two different principal quantum numbers in the basis. This can be done by adding functions which are non-zero only inside a muffin-tin sphere, constructed from the radial solutions $\phi(r)$ and $\dot{\phi}(r)$ for the two different quantum numbers. In effect, such a function can change (say) a $3p$ state into a $4p$ state. This is presumably the best way to proceed. However, it is some effort to implement and is not available in the program at this time.

- Do a two-panel calculation. For each k point, two distinct sets of eigenstates are calculated in a "valence" and a "semicore" panel. For example, the valence states for Ti are usually taken as $4s4p3d$ but the $3p$ states are extended enough to cause problems. The semicore panel would use the configuration $4s3p3d$, giving the narrow $3p$ bands as lowest eigenstates, duly occupied in this step. The higher-lying bands are slightly wrong since the $4p$ degree of freedoms are missing, but these are not occupied here. The valence panel uses the configuration $4s4p3d$ and produces the correct bandstructure above the semicore states. The valence states are occupied in this step. The two-panel approach helps in many cases, but often it leads to unsatisfactory results. For example, it is extremely difficult to find a suitable partitioning into valence and semicore states for $\mathrm{KNbO_4}$. Presently, the two-panel approach has not been implemented here.

- Use the FOCA. The idea is to use the frozen free-atom cores as in the standard frozen-core approximation. But because the cores spill out of the muffin-tin spheres, the contribution to the crystal density is obtained by overlapping the frozen cores. This is procedure is described next.

How FOCA works

The core states contribute to the eigenvalue sum and to the total electron density. When the core kinetic energy is separated off, the Kohn-Sham energy becomes

$$E_{\mathrm{KS}} = \sum \epsilon_n^{\mathrm{val}} - \int n_{\mathrm{out}}^{\mathrm{val}} V_{\mathrm{eff}} + U + E_{\mathrm{xc}} + T_{\mathrm{core}} \ . \qquad (5.18)$$

The core enters in the form of the kinetic energy

$$T_{\text{core}} = \sum \epsilon_i^{\text{core}} - \int n_{\text{out}}^{\text{core}} V_{\text{eff}} \tag{5.19}$$

and via the total electron density in U, E_{xc}, and the effective potential. The approximation of the FOCA is that the kinetic energy contribution is fixed as the sum of the free-atom core kinetic energies:

$$T_{\text{core}} \approx \sum_\nu T_{\text{core},\nu}^{\text{at}} \tag{5.20}$$

and that the crystal core density is the sum of the frozen free-atom cores:

$$n_{\text{core}}(\mathbf{r}) \approx \sum_\nu n_{\text{core},\nu}^{\text{at}}(\mathbf{r} - \mathbf{R}_\nu) \ . \tag{5.21}$$

If the cores are completely localized inside the muffin-tin spheres, this reduces to the well-known frozen-core approximation. But even when the cores extend to outside the spheres, the overlapped cores give a good approximation to the true crystal core density. If a system is first made self-consistent within the FOCA and then relaxed to full selfconsistency for valence and the core electrons together, the total energy change hereby is zero to first order in the relaxation density change.

The first step when implementing the FOCA is to obtain a suitable representation of the free-atom core density. In the free-atom program, the core outside the muffin-tin sphere is fitted by a single Hankel function, whereby the energy is varied to optimize the fit. This representation generally turns out to be quite accurate.

To overlap these cores in the crystal, we need smooth “pseudo” variants, in the same way as for the other density contributions. The Hankel function of the core fit is smoothed inside the muffin-tin sphere and a compensating gaussian is added to take up whatever amount of charge is still missing. Thus, the “pseudo” core is the sum of a smooth Hankel and a gaussian, whereby only the Hankel extends outside the muffin-tin sphere to describe the spilled-out tail. These are the functions which are overlapped and added to the smooth mesh valence density in the smooth part of the problem. Later on, the difference between true and smooth free-atom core will be added within each sphere. This is all directly analogous to the evaluation of the energy integrals as described in Section 3.2.

For the smooth density made in this way, we must calculate the electrostatic and exchange-correlation energies. It turns out that the electrostatic term is easy, because integrals over the unit cell of the type

$$\int \frac{F_1(\mathbf{r}) F_2(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \, d\mathbf{r} \, d\mathbf{r}' \tag{5.22}$$

can be done anaytically when F_1F_2 is any combination of smooth Hankels and gaussians [10].

Unexpectedly, the smooth exchange-correlation energy turns out to be the more difficult term. Even an extended semicore is still localized compared to the valence distribution. Thus, we are reluctant to represent it on the mesh designed for the valence states, since this might force us to use a finer mesh than we would otherwise. Two different solutions seem feasible:

- To obtain a smoother pseudocore, the smoothing radius is made larger, even though this begins to bend over the core density outside the muffin-tin sphere. The function is rescaled to make sure that the spilled-out charge has the right value. The gaussian term then picks up whatever charge is still left. The point is that the exact shape of the spilled-out core is not terribly critical as long as the charges inside and outside the spheres are correct. In this way, a FOCA smoothing radius of about one-half to two-thirds of R_{mt} can be used, which is smooth enough for the real-space mesh. This procedure is used if the FOCA switch on an atom is set to 1.

- Alternatively, the smooth exchange-correlation energy can be calculated to first order in the smooth core density as

$$\int \epsilon(n_0+n_1)(n_0+n_1) \approx \int \epsilon(n_0)n_0 + \int \mu(n_0)n_1 \tag{5.23}$$

  where n_0 and n_1 are the smooth valence and core densities, respectively. The advantage of this approach is that each integral involves at least one smooth factor and can be thus evaluated on the mesh. Specifically, the second integral on the right-hand side uses the smooth functions $\mu(n_0)$. Corresponding terms are subtracted inside the spheres, so that the first-order expression is only used where the n_1 is much smaller than n_0. However, this procedure seems less robust then the previous one. For example, it can go dramatically wrong if the smooth mesh density n_0 becomes slightly negative at some points, since then $\mu(n_0)$ is not smooth any more. Also, the first-order expression for E_{xc} is an additional approximation. The procedure is used if the FOCA switch is 2 on an atom.

To present experience, the best recipe seemingly is to push the FOCA smoothing radius as large as possible within the desired precision and set the FOCA switch to 1.

Using FOCA

As mentioned in the introduction to this chapter, the FOCA is a new feature. The hope is that it can replace the two-panel approach in many cases, without the need for including two principal quantum numbers simultaneously in the valence states. Theoretically, this should be possible whenever the the core spills out of the muffin-tin spheres but does not play a significant role in the electronic structure. To present experience, the recipe for using the FOCA is as follows:

- As a general rule, it is better to keep the muffin-tin sphere radii fixed when moving atoms about or changing the lattice constant. This will minimize the systematic errors connected with extended core states.
- The resulting total energy should be a good approximation to the full all-electron energy. This is in contrast to other frozen-core calculations which produce a numerically much smaller "frozen-core" energy.
- The FOCA can be switched on for each species separately, and FOCA switch settings of 1 and 2 can be used on different species in the same calculation. Presently it seems that a setting of 1 is preferable.
- If the FOCA is to be used on a species, it should be checked that the fit to the spilled-out core in the free-atom program is accurate.
- The FOCA smoothing radius should be chosen as large as possible within the desired precision. This can tested by temporarily using a finer real-space mesh to choose the radius, then reducing the mesh to acceptable convergence. If the smoothing radius can be chosen relatively large (say, one-half to two-thirds of R_{mt}), it should be possible to use a real-space mesh no finer than that required by the valence states.

Bibliography

[1] R.O. Jones and O. Gunnarsson, Rev. Mod. Phys. **61**, 689 (1989).

[2] J. Harris in *The electronic Structure of Complex Systems*, edited by W. Temmerman and P. Phariseau (Plenum, New York, 1984).

[3] M.S. Hybertsen and S. Louie, Phys. Rev. B **34**, 5390 (1986); R.W. Godby, M. Schlüter, and L.J. Sham, Phys. Rev. B **37**, 10159 (1988); F. Aryasetiawan and O. Gunnarsson, Phys. Rev. Lett. **47**, 3221 (1995).

[4] D.D. Koelling, Phys. Rev. **188**, 1049 (1969).

[5] J. Harris, Phys. Rev. B **31**, 1770 (1985); W.M.C. Foulkes and R. Haydock, Phys. Rev. B **39**, 12 520 (1989).

[6] O.K. Andersen, Phys. Rev. B **12**, 3060 (1975).

[7] J.C. Slater, Phys. Rev. **51**, 846 (1937).

[8] J. Kübler and V. Eyert, in *Materials Science and Technology* ed. K,H,J, Buschow, Vol. 3A (VCH, Weinheim, 1992).

[9] M. Methfessel, PhD thesis, Katholieke Universiteit Nijmegen (1986).

[10] Erika Bott, M. Methfessel, W. Krabs, and P.C. Schmidt, in preparation.

[11] G.B. Bachelet, D.R. Haman, and M. Schlüter, Phys. Rev. B **26**, 4199 (1982).

[12] D. Vanderbilt, Phys. Rev. B **41**, 7892 (1990).

[13] P.E. Blöchl, Phys. Rev. B **50**, 17 953.

[14] M. Methfessel and M. van Schilfgaarde, Phys. Rev. B **48**, 4937 (1993).

[15] P. Pulay, Mol. Phys. **17**, 197 (1969).

[16] A.R. Williams, J. Kübler, and C.D. Gelatt jr., Phys. Rev. B **19**, 6094 (1979).

[17] M. Methfessel, Phys. Rev. B **38**, 1537 (1988); M. Methfessel, C.O. Rodriguez, and O.K. Anderesen, Phys. Rev. B **40**, 2009 (1989).

[18] N.W. Ashcroft and N. David Mermin, *Solid State Physics*, (Holt, Rinehart and Winston, New York, 1976) p. 445.

[19] O. Jepsen and O.K. Anderesen, Solid State Commun. **9**, 1763 (1971).

[20] P.E. Blöchl, O. Jepsen, and O.K. Andersen, Phys. Rev. B **49**, 16 223 (1994).

[21] G. Kresse and J. Furthmüller, Phys. Rev. B **54**, 11 169 (1996); G. Kresse and J. Furthmüller, Comput. Mat. Sci. **6**, 15 (1996).

[22] M.J. Gillan, J. Phys. Condens. Matter **1**, 1999 (1994); A. De Vita and M.J. Gillan, J. Phys. Condens. Matter **3**, 6225 (1991).

[23] M. Methfessel and A.T. Paxton, Phys. Rev. B **40**, 3616 (1989).

[24] D. Singh, Phys. Rev. B **43**, 6388 (1991).

Index