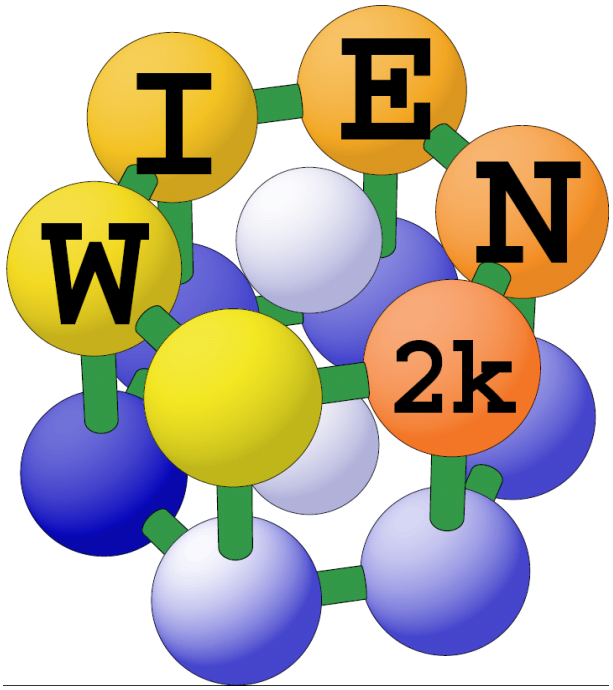

WIEN2k

An Augmented Plane Wave Plus Local Orbitals Program
for Calculating Crystal Properties

User's Guide, WIEN2k_11.1 (Release 11.04.2011)

Peter Blaha
Karlheinz Schwarz
Georg Madsen
Dieter Kvasnicka
Joachim Luitz

Vienna University of Technology
Inst. of Physical and Theoretical Chemistry
Getreidemarkt 9/156, A-1060 Vienna/Austria

Peter Blaha, Karlheinz Schwarz, Georg K. H. Madsen, Dieter Kvasnicka, Joachim Luitz:
WIEN2k
An Augmented Plane Wave + Local Orbitals Program for Calculating Crystal Properties

revised edition WIEN2k_11.1 (Release 11.04.2011)

Univ. Prof. Dr. Karlheinz Schwarz
Techn. Universität Wien
Institut für Physikalische und Theoretische Chemie
Getreidemarkt 9/156
A-1060 Wien/Austria
ISBN 3-9501031-1-2

ISBN 3-9501031-1-2

Contents

List of Tables

List of Figures

Licence conditions of WIEN2k

P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka and J. Luitz

Prof. Dr. Karlheinz Schwarz
Vienna University of Technology
Inst. of Physical and Theoretical Chemistry
A-1060 Vienna, Getreidemarkt 9/156
AUSTRIA
Fax: +43-1-58801-15698

DEFINITIONS:

In the following, the term "the authors", refers to P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka and J. Luitz at the above address. "Program" shall mean that copyrighted APW+LO code (in source and object form) comprising the computer programs known as **WIEN2k** or the graphical user interface **w2web**.

MANDATORY TERMS AND CONDITIONS:

I will adhere to the following conditions upon receipt of the program:

1. All title, ownership and rights to the program or to copies of it remain with the authors, irrespective of the ownership of the media on which the program resides.
2. I will not supply a copy of the code to anyone for any reason whatsoever. This in no way limits my making copies of the code for backup purposes, or for running on more than one computer system at my institution (it is a site license for the registered group). I will refer any request for copies of the program to the authors.
3. I will not incorporate any part of **WIEN2k** or **w2web** into any other program system, without prior written permission of the authors.
4. I will keep intact all copyright notices.
5. I understand that the authors supply **WIEN2k** and **w2web** and its documentation on an "as is" basis without any warranty, and thus with no additional responsibility or liability. I agree to report any difficulties encountered in the use of **WIEN2k** or **w2web** to the authors.
6. In any publication in the scientific literature I will reference the program as follows:

   > P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka and J. Luitz, **WIEN2k**, An Augmented Plane Wave + Local Orbitals Program for Calculating Crystal Properties (Karlheinz Schwarz, Techn. Universität Wien, Austria), 2001. ISBN 3-9501031-1-2

   Please enter your publications with WIEN2k on our web-page for "papers", so that we can easily include them in the list of WIEN-publications. In addition we like to receive a copy (ps-, pdf-file or reprint), especially for less common journals. Please send it to the second author, K. Schwarz.
7. It is understood that modifications of the **WIEN2k** or the **w2web** code can lead to problems where the authors may not be able to help. Please report useful modifications or major extensions to the authors.
8. I understand that support for running the program can not be provided in general, except on the basis of a joint project between the authors and the research partner.

1 Introduction

The Linearized Augmented Plane Wave (LAPW) method has proven to be one of the most accurate methods for the computation of the electronic structure of solids within density functional theory. A full-potential LAPW-code for crystalline solids has been developed over a period of more than twenty years. A first copyrighted version was called **WIEN** and it was published by

> P. Blaha, K. Schwarz, P. Sorantin, and S. B. Trickey, in
> Comput. Phys. Commun. 59, 399 (1990).

In the following years significantly improved and updated UNIX versions of the original **WIEN**-code were developed, which were called **WIEN93**, **WIEN95** and **WIEN97**. Now a new version, **WIEN2k**, is available, which is based on an alternative basis set. This allows a significant improvement, especially in terms of speed, universality, user-friendliness and new features.

WIEN2k is written in FORTRAN 90 and requires a UNIX operating system since the programs are linked together via C-shell scripts. It has been implemented successfully on the following computer systems: Pentium systems running under Linux, IBM RS6000, HP , SGI , Compac DEC Alpha, and SUN. It is expected to run on any modern UNIX (LINUX) system.

Hardware requirements will change from case to case (small cases with 10 atoms per unit cell can be run on any Pentium PC with 128 Mb under Linux), but generally we recommend a powerful PC or workstation with at least 256 Mb (better 512 Mb or more) memory and 1 Gb (better a few Gb) of disk space. For coarse grain parallization on the k-point level, a cluster of PCs with a 100 Mb/s network is sufficient. Faster communication is recommended for the fine grain (single k-point) parallel version.

In order to use all options and features (such as the new graphical user interface **w2web** or some of its plotting tools) the following public domain program packages in addition to a F90 compiler must be installed:

- perl 5 or higher (for **w2web** only)
- emacs or another editor of your choice
- ghostscript (with jpg support)
- gnuplot (with png support)
- www-browser
- pdf-reader (acroread,...)
- MPI+SCALAPACK (on parallel computers only)

Usually these packages should be available on modern systems. If one of these packages is not available, it can either be installed from public domain sources (see Chapt. 11) or the corresponding configuration may be changed (e.g. using vi instead of emacs). None of the principal components of **WIEN2k** requires these packages, only for advanced features or **w2web** they are needed.

WIEN2k has the following features that are new with respect to **WIEN97**:

- due to the new APW+lo basis set it is significantly faster (up to an order of magnitude). Optimizations in the most time consuming parts of LAPW1 and LAPW2 have been made.
- iterative diagonalization (for cases with large matrices and few eigenvalues)
- beside the k-point parallelization (including heterogeneous workstation clusters) a fine grain parallelization based on MPI is also available.
- A new web-based graphical user interface **w2web** has been developed. It does NOT require an X-environment and thus **WIEN2k** can be controlled from (but not run on !) any Windows-PC. This should particularly help the novice to get acquainted with **WIEN2k** but it should be useful for the regular user as well.
- support for AFM and FSM calculations
- spin-orbit coupling, including a new $p_{1/2}$-LO for higher accuracy
- wavefunction plotting
- determination of irreducible representations
- elastic constants (cubic cases only)
- Topological analysis based on Bader's "atoms in molecules" concept
- LDA+U, orbital polarization (OP), magnetic and electric fields
- Exact-exchange and Hybrid functionals inside spheres
- new PKZB and TPSS meta-GGA functionals

The development of **WIEN2k** was made possible by support from many sources. We try to give credit to all who have contributed. We hope not to have forgotten anyone who made an important contribution for the development or the improvement of the **WIEN2k** code. If we did, please let us know (we apologize and will correct it). The main developers in addition to the authors are the following groups:

- C. Ambrosch-Draxl (Univ. Graz, Austria) and her group, optics
- T. Charpin (Paris), elastic constants
- H. Hofstaetter and O.Koch (Vienna) iterative diagonalization
- K. Jorissen (Univ.Antwerp), C.Hebert (TU Wien), telnes3
- R. Laskowski (TU Vienna), mpi-parallelization, new dstart version
- L. Marks (Northwestern Univ.): speed-up, various optimizations, geometry optimization (PORT) and new mixer (MSEC1, MSR1, MSR1a)
- R. Luke (Univ. Delaware): new mixer (MSEC1)
- P. Novák and J. Kuneš (Prague), LDA+U, SO
- C. Persson (Uppsala), irreducible representations
- M. Scheffler (Fritz Haber Inst., Berlin) and his group, forces, dstart, geometry optimization
- E. Sjöstedt and L Nordström (Uppsala, Sweden), APW+lo
- J. Sofo and J. Fuhr (Barriloche), Bader analysis
- F. Tran (Vienna), Forces for orbital potential, Hybrid-Functionals
- B. Yanchitsky and A. Timoshevskii (Kiev), sgroup

We want to thank those **WIEN97** users, who reported bugs or made suggestions and thus contributed to new versions as well as persons who have made major contributions in the development of previous versions of the code:

- R. Augustyn (Vienna), U. Birkenheuer (Munich, wavefunction plotting), P. Blöchl (IBM Zürich), F. Boucher (Nantes), A. Chizmeshsya (Arizona), R.Dohmen and J.Pichlmeier (RZG Garching, parallelization) P. Dufek (Vienna), H. Ebert (Munich), E. Engel (Frankfurt), H. Enkisch (Dortmund), M. Fähnle (MPI Stuttgart), B. Harmon (Ames, Iowa), S. Kohlhammer (Stuttgart), T. Kokalj (Ljubljana), H. Krimmel (Stuttgart), P. Louf (Vienna), I. Mazin (Washington), M. Nelhiebel (Vienna), V. Petricek (Prague), C. Rodrigues (La Plata, Argentina), P. Schattschneider (Vienna), R. Schmid (Frankfurt), D. Singh (Washington), H. Smolinski (Dortmund), T. Soldner (Leipzig), P. Sorantin (Vienna), S. Trickey (Gainesville), S. Wilke (Exxon, USA), B. Winkler (Kiel)

This work was supported by the following institutions:

- Austrian Science Foundation (FWF-Projects P5939, P7063, P8176, SFB08-11)
- Siemens Nixdorf (WIEN93)
- IBM (WIEN)

We take this opportunity to thank for all contributions.
For suggestions or bug reports please contact the authors by email:

pblaha@theochem.tuwien.ac.at
kschwarz@theochem.tuwien.ac.at

Part I

Introduction to the WIEN2k package

2 The basic concepts of the present band theory approach

2.1 The density functional theory

An efficient and accurate scheme for solving the many-electron problem of a crystal (with nuclei at fixed positions) is the local spin density approximation (LSDA) within density functional theory (Hohenberg and Kohn 64, Kohn and Sham 65). Therein the key quantities are the spin densities $\rho_\sigma(r)$ in terms of which the total energy is

$$E_{tot}(\rho_\uparrow, \rho_\downarrow) = \quad T_s(\rho_\uparrow, \rho_\downarrow) + E_{ee}(\rho_\uparrow, \rho_\downarrow) + \quad E_{Ne}(\rho_\uparrow, \rho_\downarrow) + E_{xc}(\rho_\uparrow, \rho_\downarrow) + E_{NN}$$

with E_{NN} the repulsive Coulomb energy of the fixed nuclei and the electronic contributions, labelled conventionally as, respectively, the kinetic energy (of the non-interacting particles), the electron-electron repulsion, nuclear-electron attraction, and exchange-correlation energies. Two approximations comprise the LSDA, i), the assumption that E_{xc} can be written in terms of a local exchange-correlation energy density μ_{xc} times the total (spin-up plus spin-down) electron density as

$$E_{xc} = \int \mu_{xc}(\rho_\uparrow, \rho_\downarrow) * [\rho_\uparrow + \rho \downarrow] dr \tag{2.1}$$

and ii), the particular form chosen for that μ_{xc}. Several forms exist in literature, we use the most recent and accurate fit to the Monte-Carlo simulations of Ceperly and Alder by Perdew and Wang 92. E_{tot} has a variational equivalent with the familiar Rayleigh-Ritz principle. The most effective way known to minimize E_{tot} by means of the variational principle is to introduce orbitals χ^σ_{ik} constrained to construct the spin densities as

$$\rho_\sigma(r) = \sum_{i,k} \rho^\sigma_{ik} |\chi^\sigma_{ik}(r)|^2 \tag{2.2}$$

Here, the ρ^σ_{ik} are occupation numbers such that $0 \leq \rho^\sigma_{ik} \leq 1/w_k$, where w_k is the symmetry-required weight of point k. Then variation of E_{tot} gives the Kohn-Sham equations (in Ry atomic units),

$$[-\nabla^2 + V_{Ne} + V_{ee} + V^\sigma_{xc}]\chi^\sigma_{ik}(r) = \epsilon^\sigma_{ik}\chi^\sigma_{ik}(r) \tag{2.3}$$

which must be solved and thus constitute the primary computational task. This Kohn-Sham equations must be solved self-consistently in an iterative process, since finding the Kohn-Sham orbitals requires the knowledge of the potentials which themselves depend on the (spin-) density and thus on the orbitals again.

Recent progress has been made going beyond the LSDA by adding gradient terms of the electron density to the exchange-correlation energy or its corresponding potential. This has led to the generalized gradient approximation (GGA) in various parameterizations, e.g. the one by Perdew et al 92 or Perdew, Burke and Ernzerhof (PBE) 96, which is the recommended option.

A recent version called meta-GGA by Perdew et al (1999) and Tao et al. (2003) employes for the evaluation of the exchange-correlation energy not only the gradient of the density, but also the kinetic energy density $\tau(r)$. Unfortunately, such schemes are not yet self-consistent.

2.2 The Full Potential APW methods

Recently, the development of the Augmented Plane Wave (APW) methods from Slater's APW, to LAPW and the new APW+lo was described by Schwarz et al. 2001.

2.2.1 The LAPW method

The linearized augmented plane wave (LAPW) method is among the most accurate methods for performing electronic structure calculations for crystals. It is based on the density functional theory for the treatment of exchange and correlation and uses e.g. the local spin density approximation (LSDA). Several forms of LSDA potentials exist in the literature , but recent improvements using the generalized gradient approximation (GGA) are available too (see sec. 2.1). For valence states relativistic effects can be included either in a scalar relativistic treatment (Koelling and Harmon 77) or with the second variational method including spin-orbit coupling (Macdonald 80, Novák 97). Core states are treated fully relativistically (Desclaux 69).

A description of this method to linearize Slater's old APW method (i.e. the LAPW formalism) and further programming hints are found in many references: Andersen 73, 75, Koelling 72, Koelling and Arbman 75, Wimmer et al. 81, Weinert 81, Weinert et al. 82, Blaha and Schwarz 83, Blaha et al. 85, Wei et al. 85, Mattheiss and Hamann 86, Jansen and Freeman 84, Schwarz and Blaha 96). An excellent book by D. Singh (Singh 94) describes all the details of the LAPW method and is highly recommended to the interested reader. Here only the basic ideas are summarized; details are left to those references.

Like most "energy-band methods", the LAPW method is a procedure for solving the Kohn-Sham equations for the ground state density, total energy, and (Kohn-Sham) eigenvalues (energy bands) of a many-electron system (here a crystal) by introducing a basis set which is especially adapted to the problem.

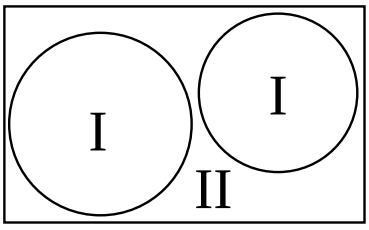

Figure 2.1: Partitioning of the unit cell into atomic spheres (I) and an interstitial region (II)

This adaptation is achieved by dividing the unit cell into (I) non-overlapping atomic spheres (centered at the atomic sites) and (II) an interstitial region. In the two types of regions different basis sets are used:

1. (I) inside atomic sphere t, of radius R_t, a linear combination of radial functions times spherical harmonics $Y_{lm}(r)$ is used (we omit the index t when it is clear from the context)
$$\phi_{\mathbf{k}_n} = \sum_{lm} [A_{lm,\mathbf{k}_n} u_l(r, E_l) + B_{lm,\mathbf{k}_n} \dot{u}_l(r, E_l)] Y_{lm}(\hat{\mathbf{r}}) \tag{2.4}$$
where $u_l(r, E_l)$ is the (at the origin) regular solution of the radial Schroedinger equation for energy E_l (chosen normally at the center of the corresponding band with l-like character) and the spherical part of the potential inside sphere t; $\dot{u}_l(r, E_l)$ is the energy derivative of u_l evaluated at the same energy E_l. A linear combination of these two functions constitute the linearization of the radial function; the coefficients A_{lm} and B_{lm} are functions of k_n (see below) determined by requiring that this basis function matches (in value and slope) each plane wave (PW) the corresponding basis function of the interstitial region; u_l and $\dot{u}_l$ are obtained by numerical integration of the radial Schroedinger equation on a radial mesh inside the sphere.

2. (II) in the interstitial region a plane wave expansion is used
$$\phi_{\mathbf{k}_n} = \frac{1}{\sqrt{\omega}} e^{i\mathbf{k}_n \cdot \mathbf{r}} \tag{2.5}$$
where $\mathbf{k}_n = \mathbf{k} + \mathbf{K}_n$; $\mathbf{K}_n$ are the reciprocal lattice vectors and $\mathbf{k}$ is the wave vector inside the first Brillouin zone. Each plane wave is augmented by an atomic-like function in every atomic sphere.

The solutions to the Kohn-Sham equations are expanded in this combined basis set of LAPW's according to the linear variation method

$$\psi_{\mathbf{k}} = \sum_n c_n \phi_{\mathbf{k}_n} \tag{2.6}$$

and the coefficients c_n are determined by the Rayleigh-Ritz variational principle. The convergence of this basis set is controlled by a cutoff parameter $R_{mt}K_{max}$ = 6 - 9, where R_{mt} is the smallest atomic sphere radius in the unit cell and K_{max} is the magnitude of the largest K vector in equation (2.6).

In order to improve upon the linearization (i.e. to increase the flexibility of the basis) and to make possible a consistent treatment of semicore and valence states in one energy window (to ensure orthogonality) additional (k_n independent) basis functions can be added. They are called "local orbitals (LO)" (Singh 91) and consist of a linear combination of 2 radial functions at 2 different energies (e.g. at the $3s$ and $4s$ energy) and one energy derivative (at one of these energies):

$$\phi^{LO}_{lm} = [A_{lm} u_l(r, E_{1,l}) + B_{lm} \dot{u}_l(r, E_{1,l}) + C_{lm} u_l(r, E_{2,l})] Y_{lm}(\hat{r}) \tag{2.7}$$

The coefficients A_{lm}, B_{lm} and C_{lm} are determined by the requirements that ϕ^{LO} should be normalized and has zero value and slope at the sphere boundary.

2.2.2 The APW+lo method

Sjöstedt, Nordström and Singh (2000) have shown that the standard LAPW method with the additional constraint on the PWs of matching in value AND slope to the solution inside the sphere is not the most efficient way to linearize Slater's APW method. It can be made much more efficient when one uses the standard APW basis, but of course with $u_l(r, E_l)$ at a fixed energy E_l in order to keep the linear eigenvalue problem. One then adds a new local orbital (lo) to have enough variational flexibility in the radial basisfunctions:

$$\phi_{\mathbf{k}_n} = \sum_{lm} [A_{lm,\mathbf{k}_n} u_l(r, E_l)] Y_{lm}(\hat{\mathbf{r}}) \tag{2.8}$$

$$\phi_{lm}^{lo} = [A_{lm}u_l(r, E_{1,l}) + B_{lm}\dot{u}_l(r, E_{1,l})]Y_{lm}(\hat{\mathbf{r}}) \tag{2.9}$$

This new lo (denoted with lower case to distinguish it from the LO given in equ. 2.7) looks almost like the old "LAPW"-basis set, but here the A_{lm} and B_{lm} do not depend on k_n and are determined by the requirement that the lo is zero at the sphere boundary and normalized.

Thus we construct basis functions that have "kinks" at the sphere boundary, which makes it necessary to include surface terms in the kinetic energy part of the Hamiltonian. Note, however, that the total wavefunction is of course smooth and differentiable.

As shown by Madsen et al. (2001) this new scheme converges practically to identical results as the LAPW method, but allows to reduce "RKmax" by about one, leading to significantly smaller basis sets (up to 50 %) and thus the corresponding computational time is drastically reduced (up to an order of magnitude). Within one calculation a mixed "LAPW and APW+lo" basis can be used for different atoms and even different l-values for the same atom (Madsen et al. 2001). In general one describes by APW+lo those orbitals which converge most slowly with the number of PWs (such as TM 3d states) or the atoms with a small sphere size, but the rest with ordinary LAPWs. One can also add a second LO at a different energy so that both, semicore and valence states, can be described simultaneously.

2.2.3 General considerations

In its general form the LAPW (APW+lo) method expands the potential in the following form

$$V(\mathbf{r}) = \begin{cases} \sum\limits_{LM} V_{LM}(r)Y_{LM}(\hat{\mathbf{r}}) & \text{inside sphere} \\ \sum\limits_{\mathbf{K}} V_{\mathbf{K}}e^{i\mathbf{K}\cdot\mathbf{r}} & \text{outside sphere} \end{cases} \tag{2.10}$$

and the charge densities analogously. Thus no shape approximations are made, a procedure frequently called a "full-potential" method.

The "muffin-tin" approximation used in early band calculations corresponds to retaining only the $l = 0$ component in the first expression of equ. 2.10 and only the $K = 0$ component in the second. This (much older) procedure corresponds to taking the spherical average inside the spheres and the volume average in the interstitial region.

The total energy is computed according to Weinert et al. 82.

Rydberg atomic units are used except internally in the atomic-like programs (LSTART and LCORE) or in subroutine outwin (LAPW1, LAPW2), where Hartree units are used. The output is always given in Rydberg units.

The forces at the atoms are calculated according to Yu et al (91). For the implementation of this formalism in WIEN see Kohler et al (96) and Madsen et al. 2001. An alternative formulation by Soler and Williams (89) has also been tested and found to be equivalent, both in computationally efficiency and numerical accuracy (Krimmel et al 94).

The Fermi energy and the weights of each band state can be calculated using a modified tetrahedron method (Blöchl et al. 94), a Gaussian or a temperature broadening scheme.

Spin-orbit interactions can be considered via a second variational step using the scalar-relativistic eigenfunctions as basis (see Macdonald 80, Singh 94 and Novák 97). In order to overcome the problems due to the missing $p_{1/2}$ radial basis function in the scalar-relativistic basis (which corresponds to $p_{3/2}$), we have recently extended the standard LAPW basis by an additional "$p_{1/2}$-local orbital", i.e. a LO with a $p_{1/2}$ basis function, which is added in the second-variational SO calculation (Kuneš et al. 2001).

It is well known that for localized electrons (like the 4f states in lanthanides or 3d states in some TM-oxides) the LDA (GGA) method is not accurate enough for a proper description. Thus we have implemented various forms of the LDA+U method as well as the "Orbital polarization method" (OP) (see Novák 2001 and references therein). In addition you can also calculate exact-exchange inside the spheres and apply various hybrid functionals (see Tran et al. 2006 for details).

One can also consider interactions with an external magnetic (see Novák 2001) or electric field (via a supercell approach, see Stahn et al. 2000).

PROPERTIES:

The density of states (DOS) can be calculated using the modified tetrahedron method of Blöchl et al. 94.

X-ray absorption and emission spectra are determined using Fermi's golden rule and dipole matrix elements (between a core and valence or conduction band state respectively). (Neckel et al. 75, Schwarz et al 79,80)

X-ray structure factors are obtained by Fourier Transformation of the charge density.

Optical properties are obtained using the "Joint density of states" modified with the respective dipole matrix elements according to Ambrosch et al. 95, Abt et al. 94, Abt 97. and in particular Ambrosch 06. A Kramers-Kronig transformation is also possible.

An analysis of the electron density according to Bader's "atoms in molecules" theory can be made using a program by J. Sofo and J. Fuhr (2001)

3 Quick Start

Contents

We assume that **WIEN2k** is properly installed and configured for your site and that you ran **userconfig_lapw** to adjust your path and environment. (For a detailed description of the installation see chapter 11.

This chapter is intended to guide the novice user in the handling of the program package. We will use the example of TiC in the sodium chloride structure to show which steps are necessary to initialize a calculation and run a self consistent field cycle. We also demonstrate how to calculate various physical properties from these SCF data. Along the way we will give all important information in a very abridged form, so that the novice user is not flooded with information, and the experienced user will be directed to more complete information.

In this chapter we will also show, how the new graphical user interface **w2web** can be utilized to setup and run the calculations.

3.1 Naming conventions

Before we begin with our introductory example, we describe the naming conventions, to which we will adhere throughout this user's guide.

On UNIX systems the files are specified by **case.type** and it is required that all files reside in a subdirectory **./case**. Here and in the following sections and in the shell scripts which run the package themselves, we follow a simple, systematic convention for file labeling.

For the general discussion (when no specific crystal is involved), we use **case**, while for a specific case, e.g. TiC, we use the following notation:

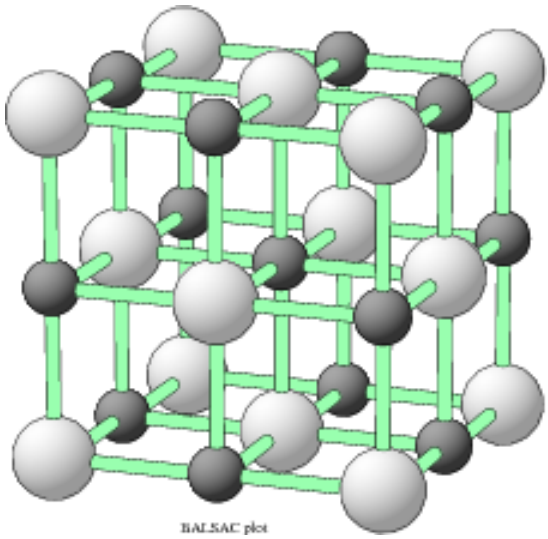

Figure 3.1: TiC in the sodium chloride structure. This plot was generated using BALSAC (see 9.23.1). Interface programs between **WIEN2k** and BALSAC are available.

case=TiC

The filetype "**type**" always describes the content of the file (e.g.,

type=inm is inPUT for mIXER).

Thus the input to MIXER for TiC is found in the file

TiC.inm

which should be in subdirectory **./TiC**.

3.2 Starting the w2web server

Start the user interface **w2web** on the computer where you want to execute **WIEN2k**(you may have to telnet, ssh,.. to this machine) with the command

w2web [-p xxxx]

If the default port (7890) used to serve the interface is already in use by some other process, you will get the error message `w2web failed to bind port 7890 - port already in use!`. Then you will have to choose a different port number (between 1024 and 65536) . Please remember this port number, you need it when connecting to the **w2web** server.

*Note: Only user **root** can specify port numbers below 1024!*

At the first startup of this server, you will also be asked to setup a username and password, which is required to connect to this server.

3.3 Connecting to the w2web server

Use your favorite WWW-browser to connect to **w2web**, specifying the correct portnumber, e.g.

```
netscape http://hostname_where_w2web_runs:7890
```

(If you do not remember the portnumber, you can find it by using "ps -ef | grep w2web" on the computer where **w2web** is running.) You should see a screen as in Fig.3.2.

3.4 Creating a new session

The user interface **w2web** uses **sessions** to distinguish between different working environments and to quickly change between different calculations. First you have to create a new session (or select an old one). Enter "TiC" and click the "*Create*" button.
Note: Creating a session does not automatically create a new directory!

You will be placed in your home directory if no working directory was designated to this session previously (or if the directory does not exist any more).

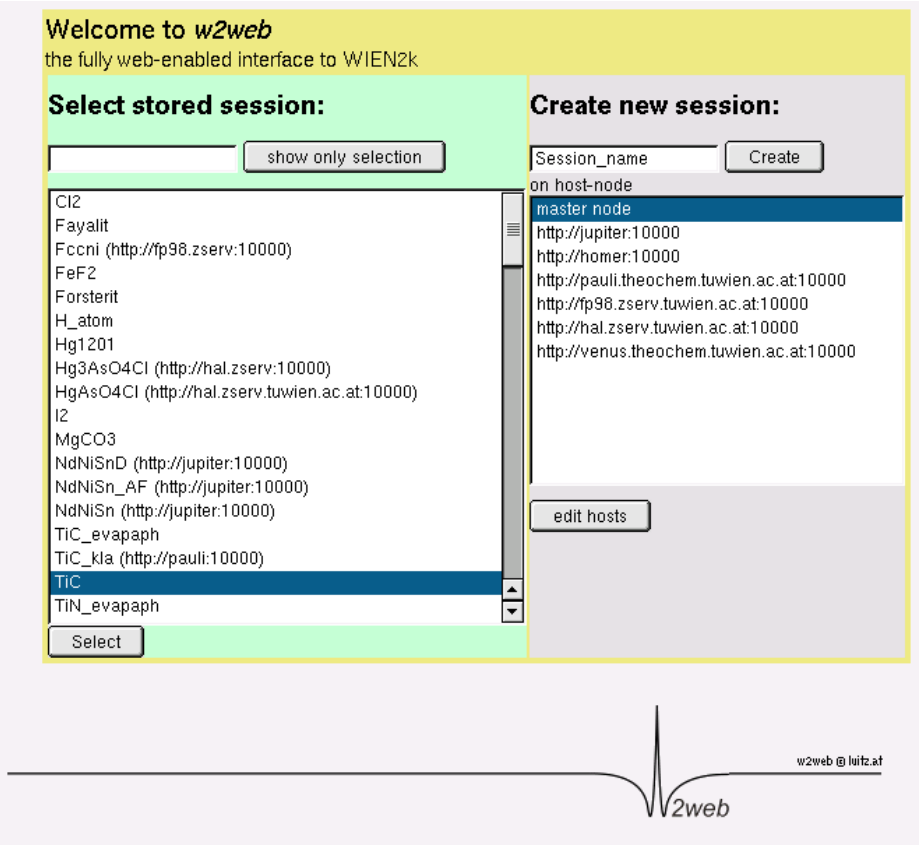

Figure 3.2: Startup screen of **w2web**

3.5 Creating a new case-directory

Using "*Session Mgmt.* ⇩ *change directory*" you can select an existing directory or create a new one. For this example create a new directory **`lapw`** and than **`TiC`** using the "*Create*" button. After the directory has been created, you have to click on *select current directory* to assign this newly created directory to the current session.

After clicking on *Click to restart session* the main window of **w2web** will appear (Fig.3.3.

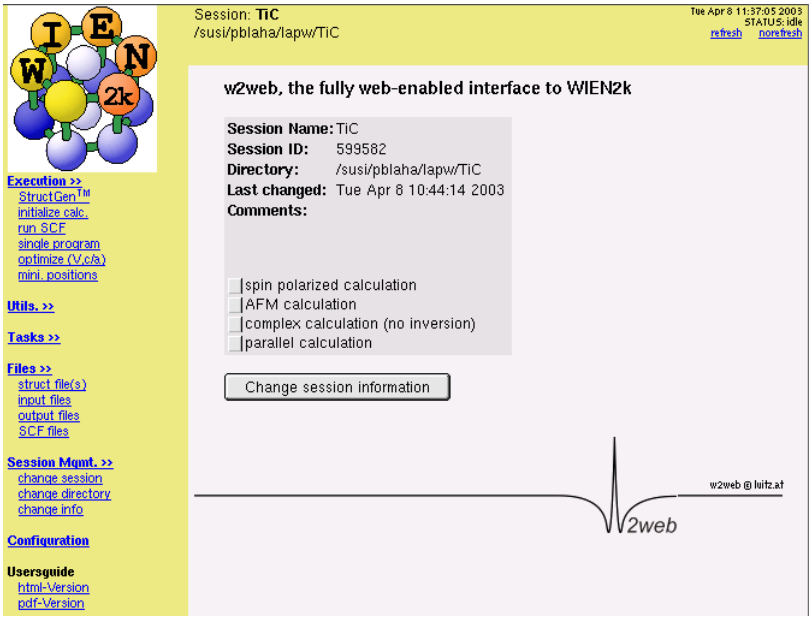

Figure 3.3: Main window of **w2web**

3.6 Creating the "master input" file case.struct

To create the file **`TiC.struct`** start the struct-file generator using "*Execution* ⇩ **StructGen**" (see figure 3.4).

For a new case **w2web** creates an empty structure template in which you can specify structural data. Later on this information is used to generate the **`TiC.struct`** file.

As a first step specify the number of atoms (2 for TiC) and fill in the data given below into the corresponding fields (white boxes):

Title	TiC
Lattice	F (for face centered)
a	4.328 Å(make sure the Ang button is selected)
b	4.328 Å
c	4.328 Å
α, β, γ	90
Atom	Ti, enter position (0,0,0)
Atom	C, enter position (.5,.5,.5)

Click **"Save Structure"** (Z will be updated automatically) and **"set automatically RMT and continue editing "**:

This will compute the nearest neigbor distances using the program **nn** and **setrmt_lapw** will then determine the optimal RMT values (muffin-tin radius, atomic sphere radius). *To learn more about the philosophy of setting RMTs see http : //www.wien2k.at/reg_user/faq*. Since it is essential to keep RMTs constant within a series of calculations (eg. when you do a Volume-optimization, see 3.11.6), you should already now decide whether you want to do just one single calculation with fixed structural parameters, or whether you intend a relaxation of internal parameters (using forces and **min_lapw**) or a volume optimization, which would required reduced RMT values.

Choose a reduction of 3 % so that we can later optimize the lattice parameter.

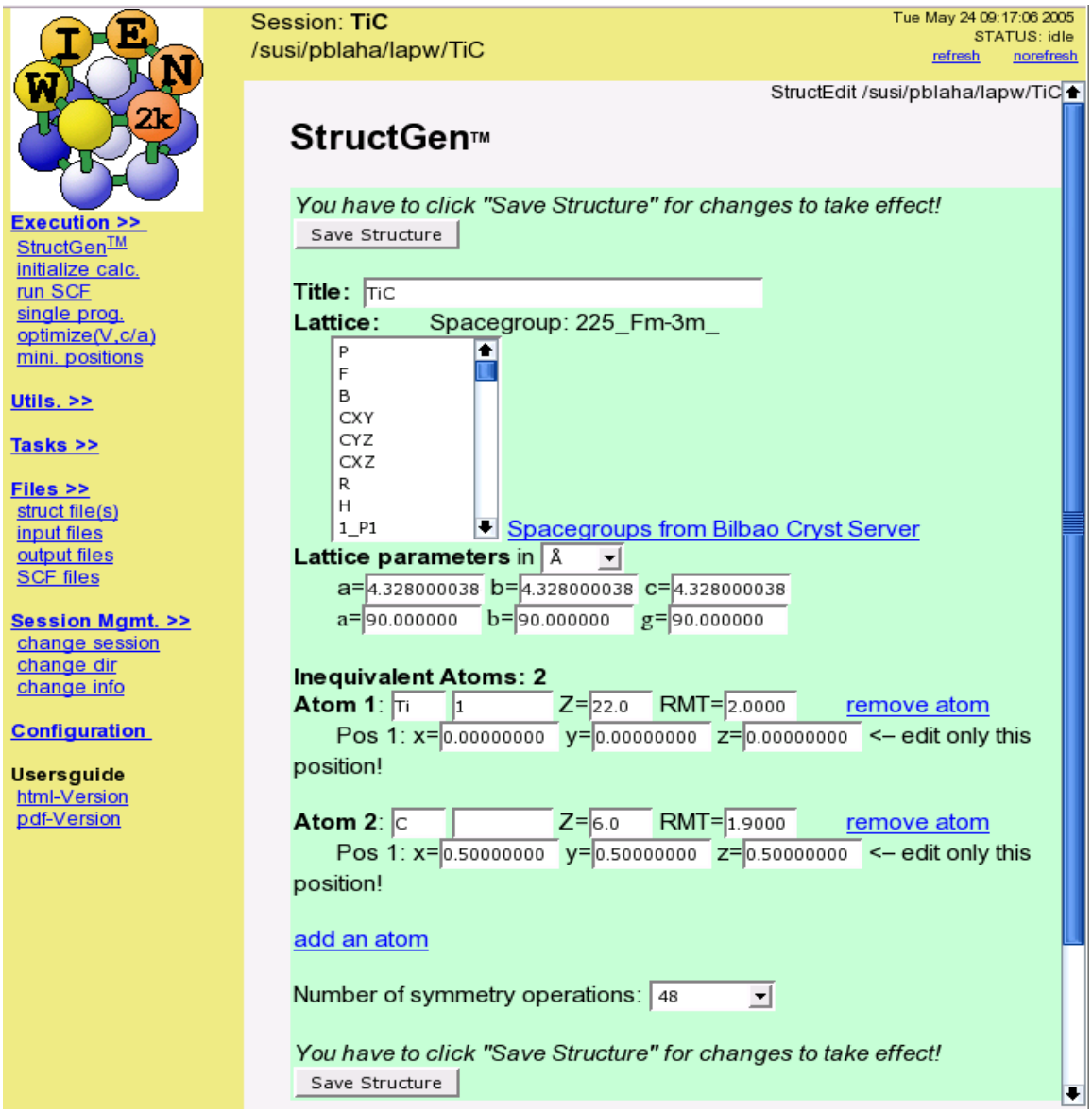

Figure 3.4: **StructGen** of **w2web**

When you are done, exit the **StructGen** with *"save file and clean up"*. This will generate the file **TiC.struct** (shown now in view-only mode with a different background color), which is the master input file for all subsequent programs.

A few other hints on **StructGen**:

You have to click on **Save Structure** after every modifications you make in the white fields. Add/remove a position/atom only if you have made no other changes before.

In a face-centered (body-centered) spacegroup you have to enter just one atom (not the ones in (.5,.5,0),...).

StructGen offers a built in calculator: Each position of equivalent atoms can be entered as a number, a fraction (e.g. $1/3$) or a simple expression (e.g. $0.21 + 1/3$). The first position defines the variables x, y and z, which can be using in expression defining the other positions (e.g. $-y$, x, $-z + 1/2$).

When you now choose "*Files ⇩ show all files*", you will see, that **tic.struct** has been created.

For a detailed description of these files consult sections 4.3 and 6.4.3.

3.7 Initialization of the calculation (init_lapw)

After the two basic input files have been created, initalization of the calculation is done by "*Execution ⇩ initialize calc.*". This will guide you through the steps necessary to initialite the calculation. Simply follow the steps that are highlighted in green and follow the instructions.

The initialization process is described in detail in section 5.1.2.

Alternatively you could run the script **init_lapw** from the command line. All actions of this script are logged in short in **:log** and in detail in the file **case.dayfile**, which can easily be accessed by *Utils. ⇩ show dayfile*.

Initializing the calculation will run several steps automatically, where **x** is the script to start **WIEN2k** programs (see section: 5.1.1).

x nn calculates the nearest neighbors up to a specified distance and thus helps to determine the atomic sphere radii (you must specify a distance factor f, e.g. 2, and all distances up to f * NN-dist. are calculated)

view TiC.outputnn : check for overlapping spheres, coordination numbers and nearest neighbor distances, (e.g. in the sodium chloride structure there must 6 nearest and 12 next nearest neighbors). Using these distances and coordinations you can check whether you put the proper positions into your struct file or if you made a mistake. **nn** also checks whether your equivalent atoms are really crystallographically equivalent and eventually writes a new struct-file which you may or may not accept. If you have not done so at the very beginning, go back to **StructGen** and choose proper RMT values. *You can save a lot of CPU-time by changing RMT to almost touching spheres. See Sec.4.3*

x sgroup calculates the point and spacegroups for the given structure

view TiC.outputsgroup : Now you can either accept the **TiC.struct** file generated by **sgroup** (if you want to use the spacegroup information or a different cell has been found by **sgroup**) or keep your original file (default).

x symmetry generates from a raw **case.struct** file the space group symmetry operations, determines the point group of the individual atomic sites, generates the LM expansion for the lattice harmonics (in **case.in2_st**) and local rotation matrices (in **case.struct_st**).

view TiC.outputs : check the symmetry operations (they have been written to or compared with already available ones in **TiC.struct** by the program symmetry) and the point group symmetry of the atoms (You may compare them with the "International Tables for X-Ray Crystallography"). If the output does not match your expectations from the "Tables", you might have made an error in specifying the positions. The **TiC.struct** file will be updated with symmetry operations, positive or negativ atomic counter (for "cubic" point group symmetries) and the local rotation matrix.

instgen_lapw : You are requested to generate an input file **TiC.inst** and can define the spin-polarization of each atom. While this is not important for TiC, it is very important for spin-polarized calculations and in particular for anti-ferromagnetic cases, where you should "flip" the spin of the AFM atoms and/or set the spin of the "non-magnetic" atoms (eg. oxygen in NiO) to zero.

x lstart generates atomic densities (see section 6.4) and determines how the orbitals are treated in the band structure calculations (i.e. as core or band states, with or without local orbitals, ...). You are requested to specify the desired exchange correlation potential and an energy that separates valence from core states. For TiC select the recommended potential option "GGA of Perdew-Burke-Ernzerhof 96" and a separation energy of -6.0 Ry.

edit TiC.outputst : check the output (did you specify a proper atomic configuration, did lstart converge, are the core electrons confined to the atomic sphere?). Warnings for the radial mesh can usually be neglected since it affects only the atomic total energy. **lstart** generates **TiC.in0_st**, **in1_st**, **in2_st**, **inc_st** and **inm_st**. For Ti it selects automatically $1s$, $2s$, and $2p$ as core states, $3s$ and $3p$ will be treated with local orbitals together with $3d$, $4s$ and $4p$ valence states.

edit TiC.in1_st : As mentioned, the input files are generated automatically with some default values which should be a reasonable choice for most cases. Nevertheless we highly recommend that you go through these inputs and become familiar with them. The most important parameter here is RKMAX, which determines the number of basis functions (size of the matrices). Values between 5-9 (APW) and 6-10 (LAPW) are usually reasonable. You may change here the usage of APW or LAPW (set 1 or 0 after the CONT/STOP switch), since often APW is necessary only for orbitals more difficult to converge (3d, 4f). *Here we will just change EMAX of the energy window from 1.5 to 2.0 Ry in order to be able to calculate the unoccupied DOS to higher energies.*

edit TiC.in2_st : Here you may limit the LM expansion (for some speedup), change the value of GMAX (in cases with small spheres (e.g. systems with H-atoms) values of 15-24 are recommended) or specify a different BZ-integration method to determine the Fermi energy. For this example you should not change anything so that you can compare your results with the test run.

Copy all generated inputs (from **case.in*_st** to **case.in***). In cases without inversion symmetry the files **case.in1c, in2c** are produced.

x kgen generates a k-mesh in the Brillouin zone (BZ). You must specify the number of k-points in the whole BZ (use 1000 for comparison with the provided output, a "good" calculations needs many more). For details see section 6.5.

view TiC.klist : check the number of k-points in the irreducible wedge of the BZ (IBZ) and the energy interval specified for the first k-point. You can now either rerun **kgen** (and generate a different k-mesh) or continue.

x dstart generates a starting density for the SCF cycle by superposition of atomic densities generated in **lstart**. For details see section 6.6.

view TiC.outputd (check if gmax >gmin)

Now you are asked , whether or not you want to run a spin-polarized calculation (in such a case case **dstart** is re-run to generate spin-densities). For TiC say **No**.

Alternatively, **w2web** provides an "expert-mode", where some inputs can be specified right at the beginning and then the whole initialization runs at once. **Please check carefully the STDOUT-listing and some output-files for possible errors or warnings!!**

Initialization of a calculation (running **init_lapw**) will create all inputs for the subsequent SCF calculation choosing some default options and values. You can find a list of input files using "*Files ⇩ input files*" (3.5).

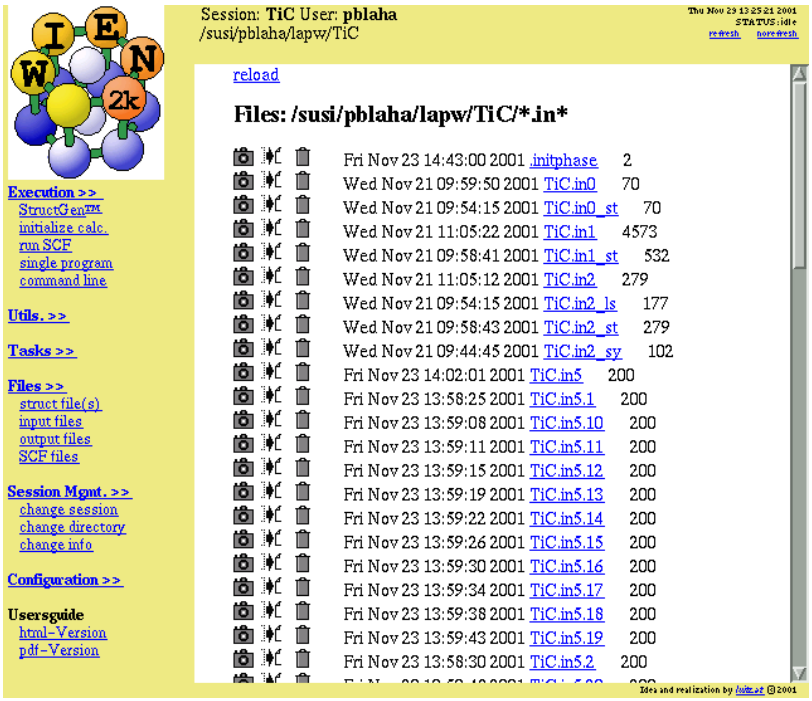

Figure 3.5: List of input files

3.8 The SCF calculation

After the case has been set up, a link to "run SCF" is added, ("*Run Programs* ⇨ *run SCF*" and you should invoke the self-consistency cycle (SCF). This runs the script **`run_lapw`** with the desired options.

The SCF cycle consists of the following steps:

LAPW0 (POTENTIAL) generates potential from density
LAPW1 (BANDS) calculates valence bands (eigenvalues and eigenvectors)
LAPW2 (RHO) computes valence densities from eigenvectors
LCORE computes core states and densities
MIXER mixes input and output densities

After selecting "run SCF" from the "Execution" menu, the SCF-window will open, and you can now specify additional parameters. For this example we select charge convergence to 0.0001: Specify "*charge*" to be used as convergence criterion, and select a value of 0.0001 (-cc 0.0001).

To run the SCF cycle, click on "*Run!*"

Since this might take a long time for larger systems; you can specify the "*Execution type*" to be *batch* or *submit* (if your system is configured with a queuing system and **w2web** has been properly set up, see section 11.3).

While the calculation is running (as indicated by the status frame in the top right corner of the window), you can monitor several quantities (see section 3.9).

Once the calculation is finished (11 iterations), view **`case.dayfile`** for timing and errors and compare your results with the files in the provided example (**`TiC/case_scf`**).

For magnetic systems you would run a spin-polarized calculation with the script **runsp_lapw**. The program flow of such a calculation is described in section 4.5.2 and the script itself in section 5.1.3.

3.9 The "history" file case.scf

During the SCF cycle the essential data of each iteration are appended to the file **case.scf**, in our example **TiC.scf**. For an easier retrieval of certain quantities, the essential lines carry a label of the form **:LABEL:** which can be used to monitor these quantities during a SCF run.

The information is retrieved using the UNIX grep command or using the "*Utils.* ⇩ *analyze*" menu.

While the SCF cycle of TiC is running try to monitor e.g. the total energy (label **:ENE**) or the charge distance (label **:DIS**). The calculation has converged, when the convergence criterion is met for three subsequent iterations (compare the charge distance in the example).

For a detailed description of the various labels consult section 4.4.

3.10 Saving a calculation

Before you proceed to another calculation, you should save the results of the SCF-cycle with the **save_lapw** command, which is also described in detail in section 5.2.1. This can also be done from the graphical user interface by choosing the "*Utils.* ⇩ *save_lapw*" menu.

Save the result to this example under the name "**TiC_scf**".

You can now improve your calculation and check the convergence of the most important parameters:

- increase RKMAX and GMAX in **case.in1** and **case.in2**
- increase the k-mesh with **x kgen**
- choose a different exchange-correlation potential in **case.in0**

Then just execute another **run_lapw** using "*Execution* ⇩ *run SCF*".

3.11 Calculating properties

Once the SCF cycle has converged one can calculate various properties like Density of States (DOS), band structure, Optical properties or X-ray spectra.

For the calculation of properties (which from now on will be called "*Tasks*"). We strongly encourage the user to utilize the user interface, **w2web**. This user interface automatically supplies input file templates and shows how to calculate the named properties on a step by step basis.

3.11.1 Electron density plots

Select "*El. Dens.*" from the "*Tasks*" menu and click on the buttons one by one (see figure 3.6):

- The total charge density includes the Ti 3s and 3p states and the resulting density around Ti would be very large and dominated by these semicore states. To get a "meaningful" picture of the chemical bonding effects one must remove these states. Inspection of **TiC.scf1** and **TiC.scf2** should allow you to select an EMIN value to eliminate the Ti 3s and 3p semicore states.

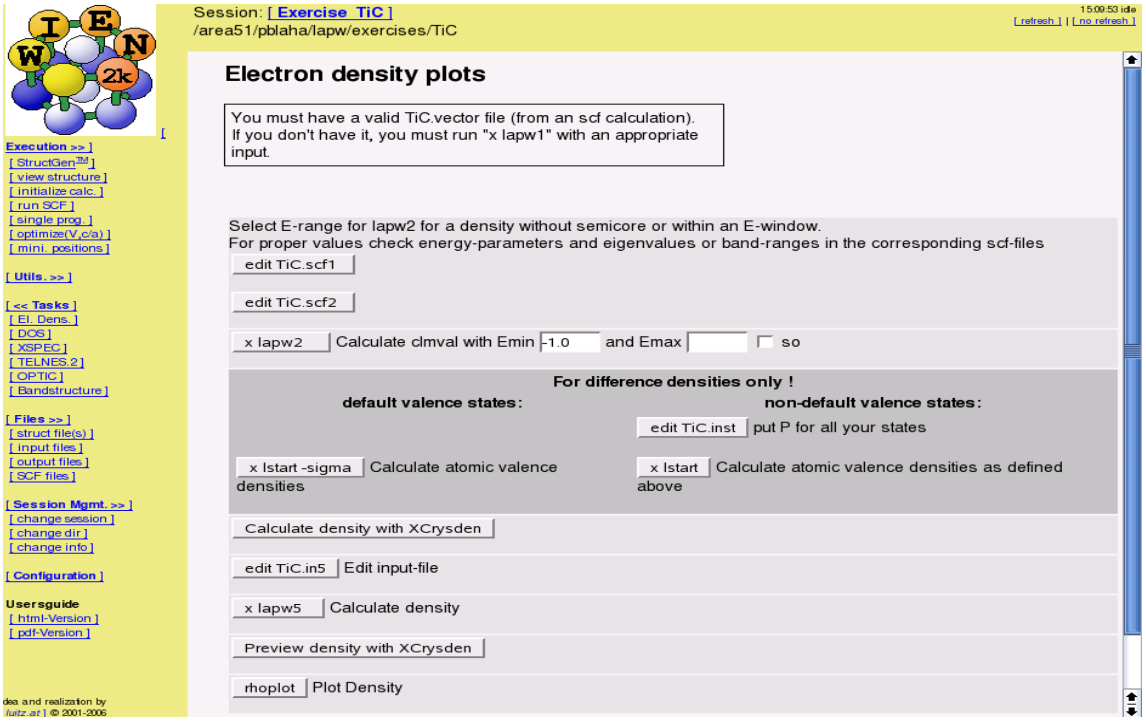

Figure 3.6: Task "Electron Density Plots"

- Recalculate the valence density with EMIN=-1.0 to truncate Ti 3s and 3p (**x lapw2**). This is only possible, when you still have a valid **TiC.vector** file on a tetrahedral mesh.
- Select a plane and plot the density in the (100) plane of TiC. When XCRYSDEN is installed (for details see http://www.xcrysden.org/doc/wien.html), it will be offered automatically and provides a convenient way to specify a plane and create a colorful plot 3.7.
  - Select 2D-plot
  - Specify a resolution of 100 points (first line)
  - Select a plane by selecting 3 atoms and define these 3 atoms by clicking on them.
  - Choose rectangular parallelogram and enlarge the rectangular selection by 0.5 (for all 4 margins, then update the display)
  - calculate the density and produce a nice contour plot:
  - choose "rainbow"-colors, activate all display-option buttens, and choose in "Ranges" a smaller "highest rendered value".
  - Finally, use smaller spheres (pipe+ball display model) and thinner bonds (Modify/Ball-Stick-ratio).
- Alternatively, without XCRYSDEN, edit **TiC.in5** and choose the offered template input file. To select the (100) plane for plotting specify the following input:

```
-1 -1  0  4      # origin of plot (x,y,z,denominator)
-1  3  0  4      # x-end of plot
 3 -1  0  4      # y-end of plot
3 2 3            # x,y,z number of shells
100 100          # x, y plotting mesh, choose ratio similar to x,y length
RHO
ANG VAL NODEBUG
```

```
ORTHO
```

For a detailed description of input options consult section 8.6.3

- Calculate electron density (x lapw5)
- Plot output (using rhoplot), after the first preview select a range zmin=-0.5 to zmax=2

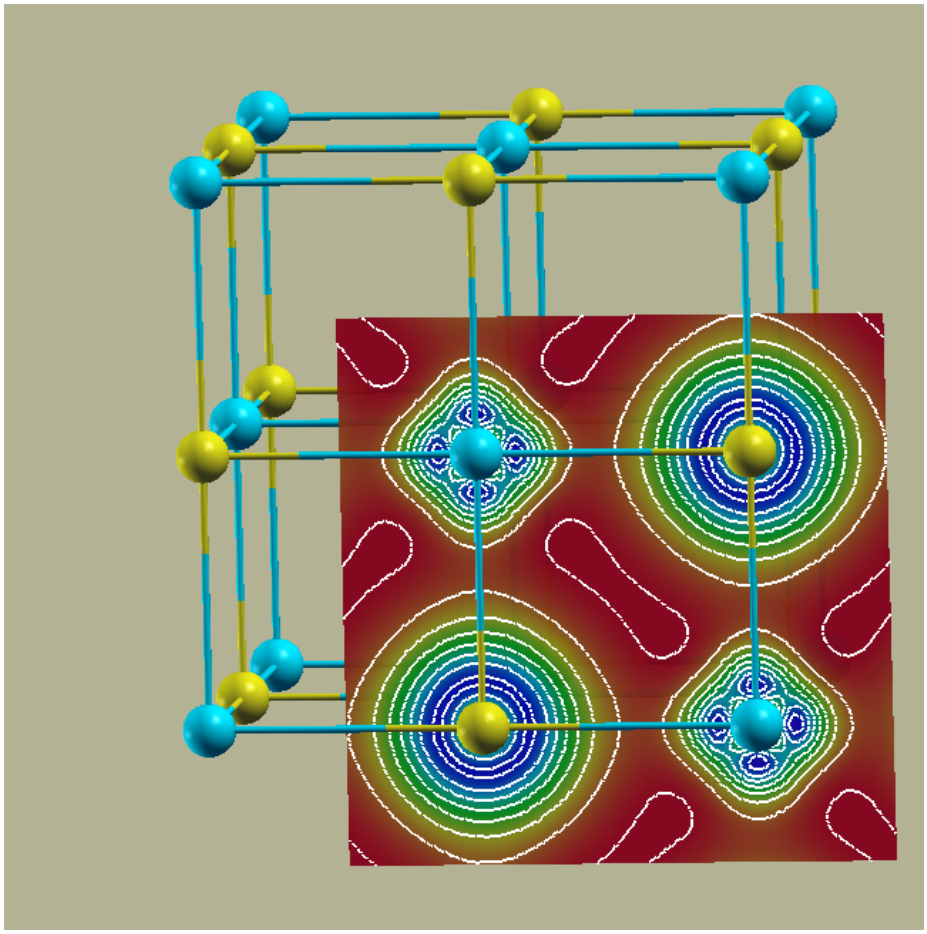

Figure 3.7: Electron density of TiC in (100) plane using Xcrysden

Compare the result with the electron density plotted in the (100) plane (see figure 3.8). The program **gnuplot** (public domain) must be installed on your computer. For more advanced graphics use your favorite plotting package or specify other options in **gnuplot** (see **`rhoplot_lapw`** how gnuplot is called).

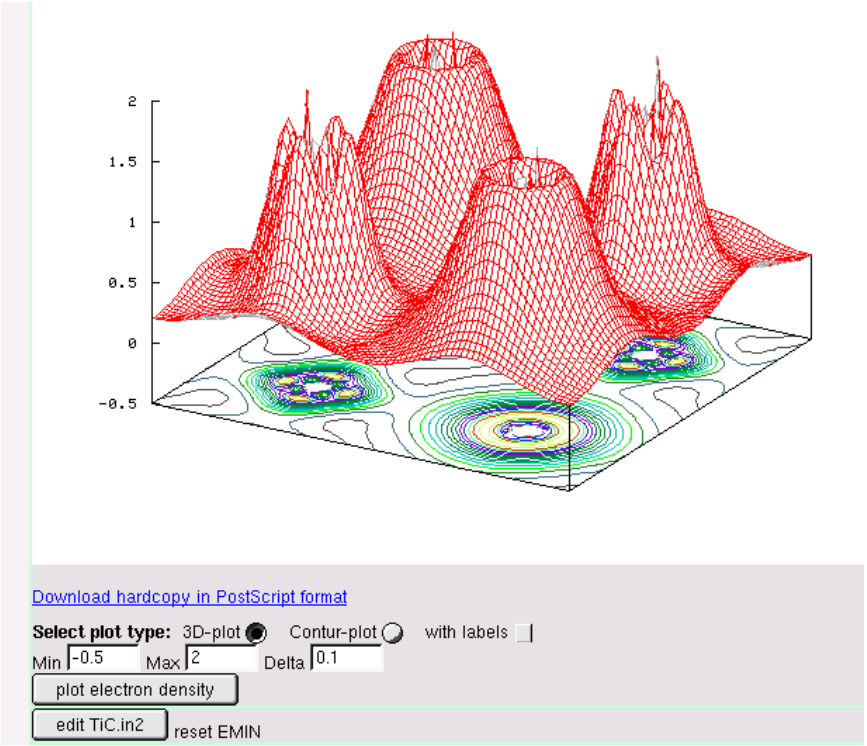

Figure 3.8: Electron density of TiC in (100) plane

3.11.2 Density of States (DOS)

Select "*Density of States (DOS)*" from the "*Tasks*" menu and click on the buttons one by one:

- Calculate partial charges (**`x lapw2 -qtl`**). (This is only possible, when you still have a valid **`TiC.vector`** file on a tetrahedral mesh.)
- Create **`TiC.int`**, either using "configure TiC.int" or/and by "editing" the offered template input file. Select: total DOS, Ti-d, Ti-d_{eg}, Ti-d_{t2g}, C-s and C-p-like DOS.

```
TiC
  -0.50     0.00200   1.500 0.003   EMIN, DE, EMAX, Gauss-broadening
    6                               NUMBER OF DOS-CASES
    0    1   tot                   (atom,case,description)
    1    4   Ti d
    1    5   Ti eg
    1    6   Ti t2g
    2    2   C s
    2    3   C p
```

  For a detailed description of input options consult section 8.1.3
- Calculate DOS (**`x tetra`**).
- Preview output using "dosplot"

If you want to use the supplied plotting interface **dosplot2** to preview the results, the program **gnuplot** (public domain) must be installed on your computer.

The calculated DOS can be compared with figures 3.9 and 3.10. Together with the electron density the partial DOS allows you to analyse the chemical bonding (covalency between $Ti-d_{eg}$ and $C-p$, non-bonding $Ti-d_{t2g}$, charge transfer estimates,....)

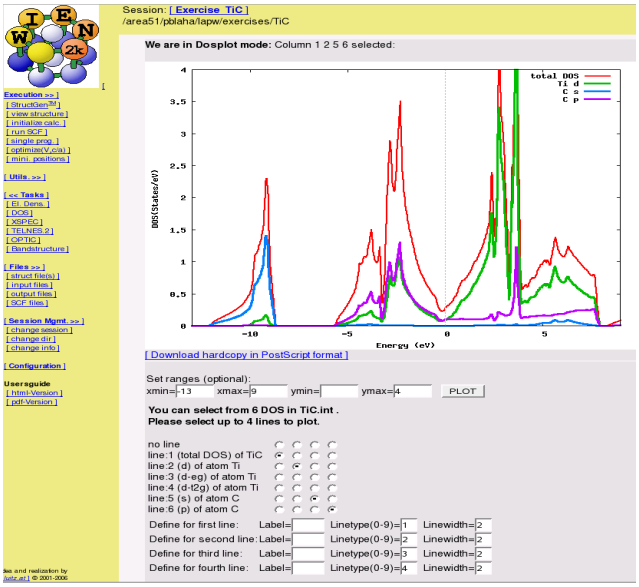

Figure 3.9: Density of states of TiC

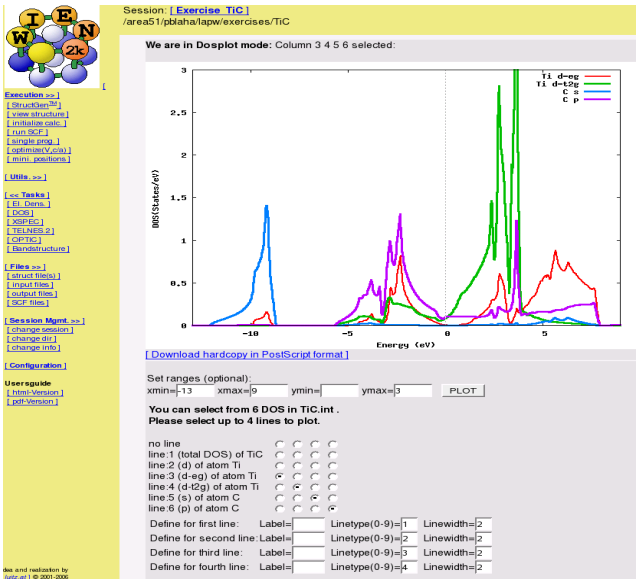

Figure 3.10: Density of states of TiC

3.11.3 X-ray spectra

Select "X-Ray Spectra" from the "Tasks menu" and click on the buttons one by one:

- Calculate partial charges (**`x lapw2 -qtl`**). This is only possible, when you still have a valid **`TiC.vector`** file on a tetrahedral mesh. *To reproduce this figure you will have to increase the EMAX value in your **`TiC.in1`** to 2.5 Ry and rerun **`x lapw1`***
- Edit **`TiC.inxs`**; choose the offered template. This template will calculate the L_{III}-spectrum of the first atom (Ti in this example) in the energy range between -2 and 15 eV. For a detailed description of the contents of this input file refer to section 8.10.3.
- Calculate spectra
- Preview spectra

If you want to use the supplied plotting interface **specplot** to preview the results, the public domain program **gnuplot** must be installed on your computer. The calculated TiC Ti-L_{III}-spectrum can be compared with figure 3.11.

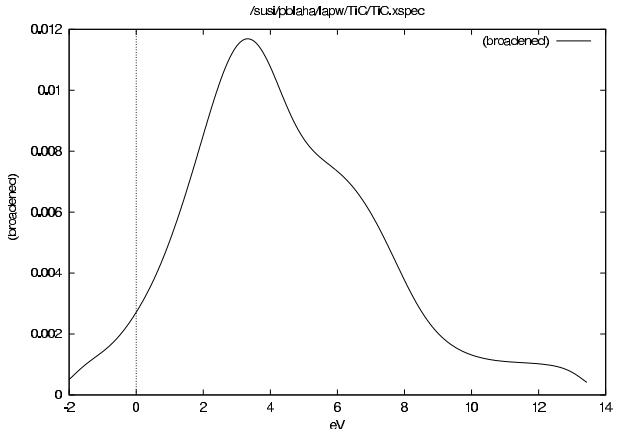

Figure 3.11: Ti L_{III} spectrum of TiC

3.11.4 Bandstructure

Select "*Bandstructure*" from the "*Tasks*" menu and click on the buttons one by one:

- Create the file **`TiC.klist_band`** from the template in **`$WIENROOT/SRC_templates/fcc.klist`**. (To calculate a bandstructure a special k-mesh along high symmetry directions is necessary. For a few crystal structures template files are supplied in the **`SRC`**-directory, you can also use XCRYSDEN (save it as xcrysden.klist) to generate a k-mesh or type in your own mesh.
- Calculate Eigenvalues using the "-band" switch (which changes **`lapw1.def`** such that the k-mesh is read from **`TiC.klist_band`** and not from **`TiC.klist`**)
  *Note: When you want to calculate DOS, charge densities or spectra after this bandstructure, you must first recalculate the **`TiC.vector`** file using the "tetrahedral" k-mesh, because the k-mesh for the band structure plots is **not** suitable for calculations of such properties.*

- Edit **TiC.insp**: insert the correct Fermi energy (which can be found in the saved **scf**-file) and specify plotting parameters. For comparison with figure 3.12 select an energy-range from -13 to 8 eV.
- Calculate Bandstructure (**x spaghetti**).
- Preview Bandstructure (needs **ghostscript installed**).

If you want to preview the bandstructure, the program **ghostview** (public domain) must be installed on your computer. You can compare your calculated bandstructure with figure 3.12.

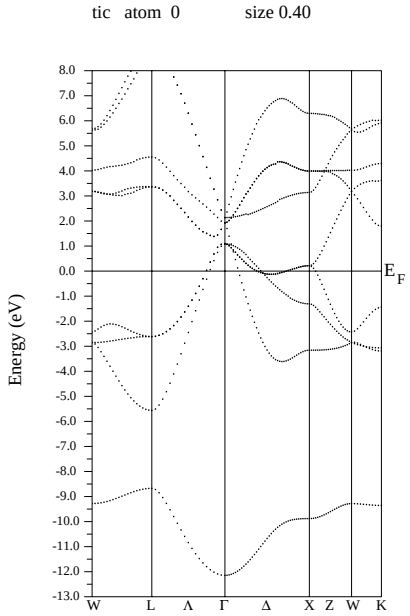

Figure 3.12: Bandstructure of TiC

3.11.5 Bandstructure with band character plotting / full lines

Select again "*Bandstructure*" from the "*Tasks*" menu. We assume that you have already done the steps described in the previous section (generate **TiC.klist_band** and **x lapw1 -band**).

- Calculate partial charges (**x lapw2 -qtl -band**)
  Note: You have to calculate the partial charges for the new special k-mesh specified above and cannot use the partial charges from the DOS calculation.
- Edit **TiC.insp**: insert the correct Fermi energy (same as before) and specify plotting parameters. For "band character plotting" (see figure 3.13) select "line type = dots" and jatom=1, jtype=6 and jsize=0.2 (in the last input line) to produce a character plot of the Ti t2g-like character bands.
- Calculate Bandstructure (**x spaghetti**)
- Preview Bandstructure
- To plot the bandstructure with full lines, calculate the irreducible representations with "x irrep" and select "lines" in **case.insp**.

*If you have **case.irrep*** or **case.qtl*** files from previous runs which do not fit to the present **case.output1** file, you may get errors while running **spaghetti**. In this case remove all **case.irrep** or **case.qtl** files.*

You can compare your results with figure 3.13.

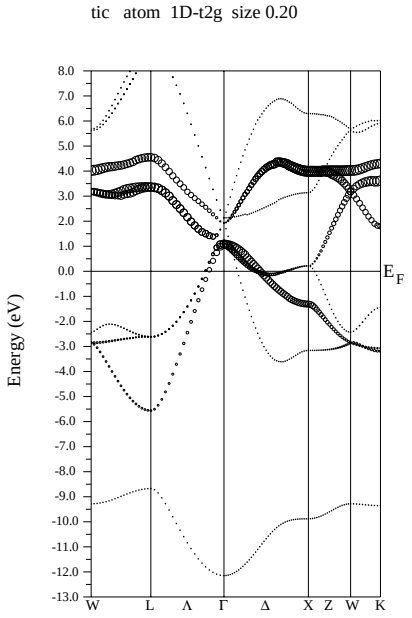

Figure 3.13: Bandstructure of TiC, showing t2g-character bands of Ti in character plotting mode

3.11.6 Volume Optimization

Select "*Optimize (V,c/a)*" from the "*Execution*" menu. Setup the shell script **`optimize.job`** script using **`x optimize`** and volume variations of -10, -5, 0, +5 and +10%. Then run the **`optimize.job`**. When the job has finished, you should click on *Plot* and then preview the energy curve.

You should get an energy curve as in figure 3.14. On the screen you will find the fitting parameters for the "equation of states" (Murnaghan, Birch-Murnaghan and the EOS2 equation, see sec. 9.12). This information is also written to **`TiC.outputeos`**.

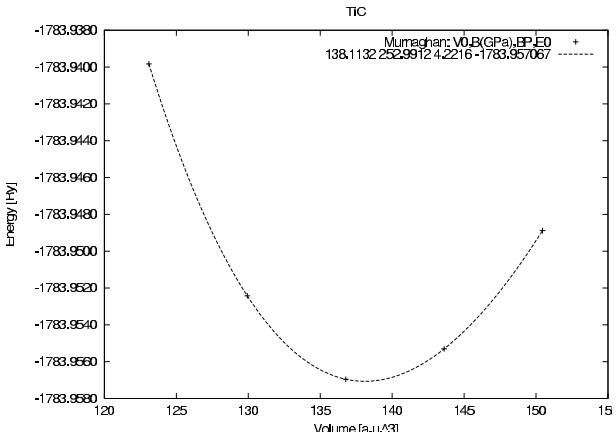

Figure 3.14: Energy vs. volume curve for TiC

3.12 Setting up a new case

In order to setup a new case you need at least the following information:

- The lattice parameters (in Bohr or Ångstroms) and angles,
- the lattice type (primitive, face-centered, hexagonal,...) or spacegroup,
- the position of all equivalent and inequivalent atoms in fractions of the unit cell.
- Alternatively with the new **StructGen** you can specify the spacegroup and only the inequivalent positions. The equivalent ones will be generated automatically.

Usually this information can be collected from the "International Tables of Crystallography" once you know the space group, the Wyckoff position and the internal free coordinates.

3.12.1 Manually setting up a new case

Usually for a new "case" the input is not created from scratch, but one uses the **struct** file from a similar case as pattern. Change into the **lapw** subdirectory and proceed as follows:

```
mkdir case_new
cd case_new
cp ../case_old/case_old.struct case_new.struct
```

Now edit **case_new.struct** (see section 4.3) as necessary (Note: this is a fixed formatted file, so all values must remain at their proper columns). Afterwards generate **case_new.inst** using **instgen_lapw**.

3.12.2 Setting up a new case using w2web

Use the menu *Session Mgmt.* ⇩ *change session* of **w2web** to create a new session (enter the name of the new session and click on "*Create*"). Then you should also create a new directory and "select" it..

When you select "*Execution* ⇩ **StructGen**", you have several choices:

You can just specify the number of non-equivalent atoms and a template file will be created. In **StructGen** you simply specify the lattice (type or spacegroup), cell parameters and name and positions of atoms. When you "*save file and clean up*" the new **case.struct** file and the **case.inst** file are created automatically.

Alternatively, you can use **cif2struct** or **xyz2struct** to convert a "**cif**", "**txt**" or "**xyz**" file into the WIEN2k **case.struct** file. Check page 168 for more info on the specific file formats.

For more information on the **StructGen** refer to page 169.

Part II

Detailed description of the files and programs of the WIEN2k package

4 File structure and program flow

Contents

(for naming conventions see section 3.1)

4.1 Flow of input and output files

Each program is started with (at least) one command line argument, e.g.

```
programX programX.def
```

in which the arguments specifies a filename, in which FORTRAN I/O units are connected to unix filenames. (See examples at specific programs). These "**def**"-files are generated automatically when the standard **WIEN2k** scripts **x**, **init_lapw** or **run_lapw** are used, but may be tailored by hand for special applications. Using the option

```
x program -d
```

a def-file can be created without running the program. In addition each program reads/writes the following files:

case.struct a "master" input file, which is described below (Section 4.3)
case.inX a specific input file, where X labels the program (see def-files for each program in chapter 6).
case.outputX an output file

The programs of the SCF cycle (see figure 4.1) write the following files:

case.scfX a file containing only the most significant output (see description below).
program.error error report file, should be empty after successful completion of a program (see chapter 6)

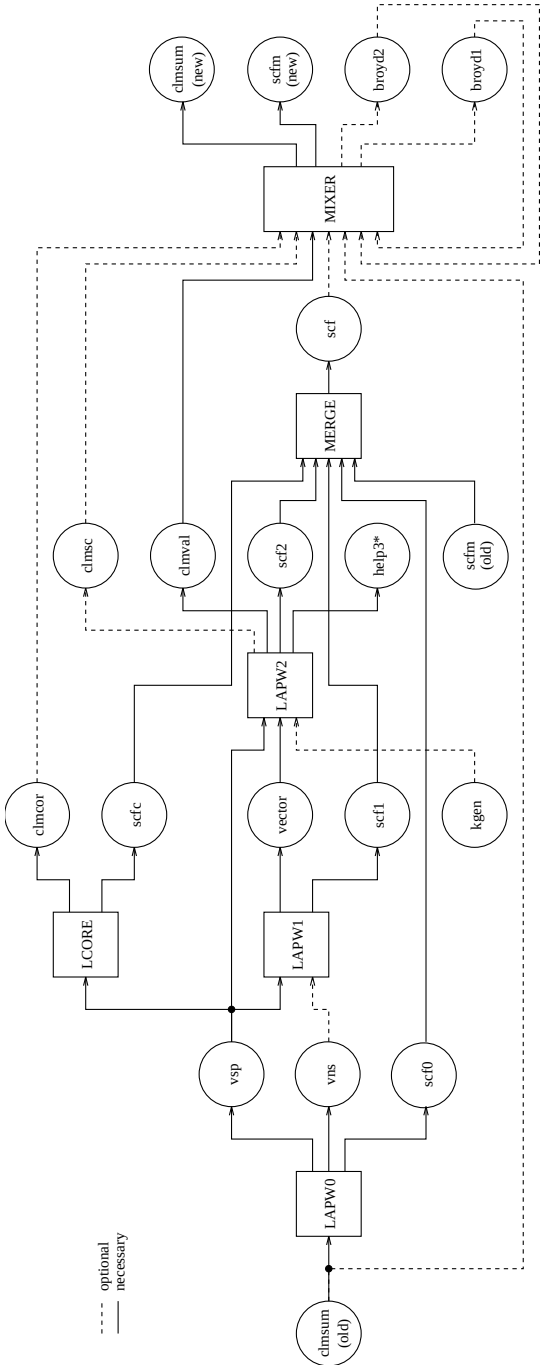

Figure 4.1: Data flow during a SCF cycle (programX.def, case.struct, case.inX, case.outputX and optional files are omitted)

The following tables describe input and output files for the initialization programs **nn**, **sgroup**, **symmetry**, **lstart**, **kgen**, **dstart** (table 4.1), the utility programs **tetra**, **irrep**, **spaghetti**, **aim**, **lapw7**, **elnes**, **lapw3**, **lapw5**, **xspec**, **optic**, **joint**, **kram**, **optimize** and **mini** (table 4.2) as well as for a SCF cycle of a non-spin-polarized case (table 4.2). Optional input and output files are used only if present in the respective case subdirectory or requested/generated by an input switch. The connection between FORTRAN units and filenames are defined in the respective **programX.def** files. The data flow is illustrated in Fig. 4.1.

program	needs		generates	
	necessary	optional	necessary	optional
NN	nn.def case.struct		case.outputnn	case.struct_nn
SGROUP	case.struct		case.outputsgroup	case.struct_sgroup
SYMMETRY	symmetry.def case.struct	 case.in2_st	case.outputs case.in2_st	case.struct_st
LSTART	lstart.def case.struct case.inst		case.outputst case.rsp case.in0_st case.in1_st case.in2_st case.inc_st case.inm_st case.inm_restart	case.rspup case.rspdn case.vsp_st case.vspdn_st case.sigma
KGEN	kgen.def case.struct		case.outputkgen case.klist case.kgen	
DSTART	dstart.def case.struct case.rsp(up) case.in0 case.in1 case.in2		case.outputd case.clmsum(up) dstart.error case.in0_std	

Table 4.1: Input and output files of init programs

program	needs		generates	
	necessary	optional	necessary	optional
SPAGHETTI	spaghetti.def case.insp case.struct case.output1	case.qtl case.outputso case.irrep	case.spaghetti_ps case.outputsp case.band.agr	case.spaghetti_ene
TETRA	tetra.def case.int case.qtl case.kgen		case.outputt case.dos1(2,3) case.dos1ev(1,2,3)	
LAPW3	lapw3.def case.struct case.in2 case.clmsum		case.output3 case.rho case.clmsum	
LAPW5	lapw5.def case.struct case.in5 case.clmval	case.sigma	case.output5 case.rho	case.rho.oned
XSPEC	xspec.def case.inc case.int case.vsp case.struct case.qtl		case.outputx case.dos1ev case.xspec case.txspec case.m1 case.m2	case.coredens
OPTIC	optic.def case.struct case.mat_diag case.inop case.vsp case.vector		case.outputop case.symmat	
JOINT	joint.def case.injoint		case.outputjoint case.joint	case.sigma_intra case.intra

continued on next page

	case.struct case.kgen case.weight case.symmat case.mat_diag			
KRAM	kram.def case.inkram case.joint		case.epsilon case.sigmak	case.eloss case.sumrules
OPTIMIZE	case.struct	case_initial.struct	optimize.job	case_vol_xxxxx.struct case_c/a_xxxxx.struct
MINI	mini.def case.inM case.finM case.scf case.struct	case.scf_mini case.tmpM case.constraint case.clmhist .min_hess	case.outputM case.tmpM1 case.struct1 case.scf_mini1 .minrestart	case.clmsum_inter
IRREP	case.struct case.vector		case.outputirrep case.irrep	
AIM	case.struct case.clmsum case.inaim		case.outputaim case.surf	case.crit
LAPW7	case.struct case.vector case.in7 case.vsp		case.output7 case.grid case.psink	case.abc
QTL	case.struct case.vector case.inq case.vsp		case.outputq case.qtl	

Table 4.2: Input and output files of utility programs

program	needs		generates	
	necessary	optional	necessary	optional
LAPW0	lapw0.def case.struct case.in0 case.clmsum	case.clmup/dn case.vrespsum/up/dn case.inm	case.output0 case.scf0 case.vsp(up/dn) case.vns(up/dn)	case.r2v case.vcoul case.vtotal
ORB	orb.def case.struct case.inorb case.dmat case.vsp	case.energy case.vorb_old	case.outputorb case.scforb case.vorb orb.error	case.br1orb case.br2orb
LAPW1	lapw1.def case.struct case.in1 case.vsp case.klist	case.vns case.vorb case.vector.old	case.output1 case.scf1 case.vector case.energy	case.nsh(s) case.nmat_only
LAPWSO	lapwso.def case.struct case.inso case.in1 case.vector case.vsp case.vns case.energy	case.vorb case.normso	case.vectorso case.outputso case.scfso case.energyso	
LAPW2	lapw2.def case.struct case.in2 case.vector case.vsp case.energy	case.kgen case.nsh case.weight case.weigh case.recprlist	case.output2 case.scf2 case.clmval	case.qtl case.weight case.weigh case.help03* case.vrespval case.almblm case.radwf
LAPWDM	lapwdm.def case.struct case.indm case.vector case.vsp case.weigh case.energy	case.inso	case.outputdm case.scfdm case.dmat lapwdm.error	
SUMPARA	case.struct case.clmval	case.scf2p	case.outputsum case.clmval	

continued on next page

			case.scf2	
LCORE	lcore.def case.struct case.inc case.vsp	case.vns	case.outputc case.scfc case.clmcor lcore.error	case.corewf
After LCORE the **case.scfX** files are appended to **case.scf** and the **case.clmsum** file is renamed to **case.clmsum_old** (see **run_lapw**)				
MIXER	mixer.def case.struct case.inm case.clmval	case.clmsum_old case.clmsc case.clmcor case.scf case.broyd1 case.broyd2	case.outputm case.scfm case.clmsum mixer.error	case.broyd*
After MIXER the file **case.scfm** is appended to **case.scf**, so that after an iteration is completed, the two essential files are **case.clmsum** and **case.scf**.				

Table 4.3: Input and output files of main programs in an SCF cycle

4.2 Description of general input/output files

In the following section the content of the (non-trivial) output files is described:

case.almblm Contains the A_{lm}, B_{lm}, C_{lm} coefficients of the wavefunctions (generated optional by **lapw2**).

case.broydX Contains the charge density of previous iterations if you use Broyden's method for mixing. They are removed when using **save_lapw**. They should be removed by hand when calculational parameters (RKMAX, kmesh, ...) have been changed, or the calculation crashed due to a too large mixing and are restarted by using a new density generated by dstart.

case.clmcor Contains the core charge density (as $\sigma(r) = 4\pi r^2 \rho(r)$ and has only a spherical part). In spin-polarized calculations two files case.clmcorup and case.clmcordn are used instead.

case.clmsc Contains the semi-core charge density in a 2-window calculation, which is no longer recommended. In spin-polarized calculations two files are used instead: **case.clmscup** and **case.clmscdn**.

case.clmsum Contains the total charge density in the lattice harmonics representation and as Fourier coefficients. (The LM=0,0 term is given as $\sigma(r) = 4\pi r^2 \rho(r)$, the others as $r^2 \rho_{LM}(r)$; suitable for generating electron density plots using **lapw5** when the TOT-switch is set, (see section 8.6). In spin-polarized calculations two additional files **case.clmup** and **case.clmdn** contain the spin densities. Generated by **dstart** or **mixer**.

case.clmval Contains the valence charge density as $r^2 \rho_{LM}(r)$; suitable for generating valence electron density plots using **lapw5** when the VAL-switch is set, (see 8.6). In spin-polarized calculations two files **case.clmvalup** and **case.clmvaldn** are used instead.

case.dmatup/dn Contains the density matrix generated by **lapwdm** for LDA+U, OP or Hybrid-DFT calculations.

case.dosX Contains the density of states (states/Ry) and corresponding energy (in Ry at the internal energy scale) generated by **tetra**. X can be 1-3. Additional files **case.dosXev** contain the DOS in (states/eV) and the energy in eV with respect to EF.

case.help03X Contains eigenvalues and partial charges for atom number X.

case.kgen This file contains the indices of the tetrahedra in terms of the list of k-points. It is used in **lapw2** (if EFMOD switch in **case.in2** is set to TETRA, see 7.5.3) and in **tetra**.

case.klist This file contains a list of k-points in the first BZ and represents a tetrahedral (special point) mesh. It is generated in **kgen** and can either be inserted into the **case.in1** file or used directly in **kgen**.

case.qtl Contains eigenvalues and corresponding partial charges (bandwise) in a form suitable for **tetra** and band structure plots with "band character". The decomposition of these charges is controlled by ISPLIT in case.struct.

case.radwf Contains the radial basis functions inside spheres (generated optional by **lapw2**).

case.rho Contains the electron densities on a grid in a specified plane generated by **lapw5**. This file can be used as input for your favorite contour or 3D plotting program.

case.rsp Contains the atomic densities generated by **lstart**. They are used by **dstart** to generate a first crystalline density (**case.clmsum**).

case.r2v Contains the exchange potential (in the lattice harmonics representation as $r^2 * V_{LM}(r)$ and as Fourier coefficients) in a form suitable for plotting with **lapw5**.

case.scf_mini Contains the last scf-iteration of each individual time (geometry) step during a structural minimization using **mini**. Thus this file contains a complete history of properties (energy, forces, positions) during a structural minimization.

case.sigma Contains the atomic densities for those states with a "P" in **case.inst**. Generated in **lstart** and used for difference densities in **lapw5**.

case.spaghetti_ps A ps file with the energy bandstructure plot generated by **spaghetti**.

case.band.agr A xmgrace file with the energy bandstructure plot generated by **spaghetti**.

case.vcoul Contains the Coulomb potential (in the lattice harmonics representation as $r^2 * V_{LM}(r)$ and as Fourier coefficients) in a form suitable for plotting with **lapw5**.

case.vorb Contains the orbital potential (in Ry) generated by **orb** for LDA+U or hybrid-DFT calculations in form of a (2l+1,2l+1) matrix.

case.vtotal Contains the total potential (in the lattice harmonics representation as $r^2 * V_{LM}(r)$ and as Fourier coefficients) in a form suitable for plotting with **lapw5**.

case.vector Binary file, contains the eigenvalues and eigenvectors of all k-points calculated in **lapw1**. In spin-polarized calculations two files **case.vectorup** and **case.vectordn** are used instead. **lapwso** generates **case.vectorso**.

case.energy Contains the eigenvalues of all k-points calculated in **lapw1**. In spin-polarized calculations two files **case.vectorup** and **case.vectordn** are used instead. **lapwso** generates **case.energyso**.

case.vns Contains the non-spherical part of the total potential V. Inside the sphere the radial coefficients of the lattice harmonics representation are listed (for L greater than 0), while for the interstitial region the reanalyzed Fourier coefficients are given (see equ. (2.10)). In spin-polarized calculations two files **case.vnsup** and **case.vnsdn** are used instead.

case.vorbup/dn Contains the orbital dependent part of the potential in LDA+U, OP or Hybrid-DFT calculations. Generated in **orb**, used in **lapw1**.

case.vsp Contains the spherical part of the total potential V stored as $r * V$ (thus the first values should be close to $-2 * Z$). In spin-polarized calculations two files **case.vspup** and **case.vspdn** are used instead.

4.3 The "master input" file case.struct

The file **case.struct** defines the structure and is the main input file used in all programs. We provide several examples in the subdirectory

example_struct_file

If you are using the "*Struct Generator*" from the graphical user interface **w2web**, you don't have to bother with this file directly! However, the description of the fields of the input mask can be found here.

Note: If you are changing this file manually, please note that this is a formatted file and the proper column positions of the characters are important! Use REPLACE instead of DELETE and INSERT during edit!

We start the description of this file with an abridged example for rutile TiO_2 (adding line numbers):

```
-------------------- top of file ----------------------line #
Titaniumdioxide TiO2 (rutile):  u=0.305                     1
P   LATTICE,NONEQUIV. ATOMS  2                              2
MODE OF CALC=RELA                                           3
 8.6817500 8.6817500 5.5916100 90.       90.       90.      4
ATOM  -1: X= 0.0000000 Y= 0.0000000 Z= 0.0000000            5
```

```
          MULT= 2          ISPLIT= 8                        6
ATOM  -1: X= 0.5000000 Y= 0.5000000 Z= 0.5000000
Titanium   NPT=  781  R0=.000022391 RMT=2.00000000   Z:22.0 7
LOCAL ROT MATRIX:    -.7071068 0.7071068 0.0000000          8
                     0.7071068 0.7071068 0.0000000          9
                     0.0000000 0.0000000 1.0000000         10
ATOM  -2: X= 0.3050000 Y= 0.3050000 Z= 0.0000000
          MULT= 4          ISPLIT= 8
ATOM  -2: X= 0.6950000 Y= 0.6950000 Z= 0.0000000
ATOM  -2: X= 0.8050000 Y= 0.1950000 Z= 0.5000000
ATOM  -2: X= 0.1950000 Y= 0.8050000 Z= 0.5000000
Oxygen     NPT=  781  R0=.000017913 RMT=1.60000000   Z: 8.0
LOCAL ROT MATRIX:    0.0000000 -.7071068 0.7071068
                     0.0000000 0.7071068 0.7071068
                     1.0000000 0.0000000 0.0000000
  16 SYMMETRY OPERATIONS:                                   11
 1 0 0  0.00                                                12
 0 1 0  0.00                                                13
 0 0 1  0.00                                                14
       1                                                    15
 1 0 0  0.00
 0 1 0  0.00
 0 0-1  0.00
       2
  ........
      15
 0 1 0  0.50
-1 0 0  0.50
 0 0 1  0.50
      16
------------------ bottom of file ---------------------------
```

Interpretive comments on this file are as follows.

P	all primitive lattices except hexagonal	[a sin(γ) sin(β), a cos(γ) sin(β), cos(β)], [0, b sin(α), b cos(α)], [0, 0, c]
F	face-centered	[a/2, b/2, 0], [a/2, 0, c/2], [0, b/2, c/2]
B	body-centered	[a/2, -b/2, c/2],[a/2, b/2, -c/2], [-a/2, b/2, c/2]
CXY	C-base-centered (orthorhombic only)	[a/2, -b/2, 0], [a/2, b/2, 0], [0, 0, c]
CYZ	A-base-centered (orthorhombic only)	[a, 0, 0], [0, -b/2, c/2], [0, b/2, c/2]
CXZ	B-base-centered (orthorh. and monoclinic symmetry)	[a sin(γ)/2, a cos(γ)/2, -c/2], [0, b, 0], [a sin(γ)/2, a cos(γ)/2, c/2]
R	rhombohedral	[a/$\sqrt{3}$/2, -a/2, c/3],[a/$\sqrt{3}$/2, a/2, c/3],[-a/$\sqrt{3}$, 0, c/3]
H	hexagonal	[$\sqrt{3}$a/2, -a/2, 0],[0, a, 0],[0, 0, c]

Table 4.4: Lattice type, description and bravais matrix used in **WIEN2k**

line 1: format (A80)
title (compound)
line 2: format (A4,23X,I3)
lattice type, NAT

lattice type	as defined in table 4.4. For definitions of the triclinic lattice see **SRC_nn/dirlat.f**
NAT	number of inequivalent atoms in the unit cell

line 3: format (13X,A4)
mode

RELA	fully relativistic core and scalar relativistic valence
NREL	non-relativistic calculation

line 4: format (6F10.6)
a, b, c, α, β, γ

a, b, c — unit cell parameters (in a.u., 1 a.u. = 0.529177 Å). In face- or body-centered structures the non-primitive (cubic) lattice constant, for rhombohedral (R) lattices the hexagonal lattice constants must be specified. (The following may help you to convert between hexagonal and rhombohedral specifications:
$a_{hex} = 2cos(\frac{\pi-\alpha_{rhomb}}{2})a_{rhomb}$
$c_{hex} = 3\sqrt{a^2_{rhomb} - \frac{1}{3}a^2_{hex}}$
and (for fcc-like lattices) $a_{rhomb} = a_{cubic}/\sqrt{2}$

α, β, γ — angles between unit axis (if omitted, 90° is set as default). Set it only for P and CXZ lattices

line 5: format (4X,I4,4X,F10.8,3X,F10.8,3X,F10.8)
atom-index, x, y, z

atom-index — running index for inequivalent atoms
positive in case of cubic symmetry
negative for non-cubic symmetry
this is set automatically using **`symmetry`**

x,y,z — position of atom in internal units, i.e. as **positive** fractions of unit cell parameters. ($0 \leq x \leq 1$; the positions in the unit cell are consistent with the convention used in the International Tables of Crystallography 64. In face- (body-) centered structures only one of four (two) atoms must be given, eg. in Fm3m position 8c is specified with 0.25, 0.25, 0.25 and .75, 0.75, 0.75). For R lattice use rhombohedral coordinates. (To convert from hexagonal into rhombohedral coordinates use the auxiliary program **`hex2rhomb`**, which can be called at a command-line:

$$\vec{X}_{ortho} = \vec{X}_{hex} \begin{pmatrix} 0 & 1 & 0 \\ \frac{\sqrt{3}}{2} & \frac{-1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\vec{X}_{rhomb} = \vec{X}_{ortho} \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{-2}{\sqrt{3}} \\ -1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

line 6: format (15X,I2,17X,I2)
multiplicity, isplit

multiplicity — number of equivalent atoms of this kind
isplit — this is just an output-option and is used to specify the decomposition of the lm-like charges into irreducible representations, useful for interpretation in case.qtl). This parameter is automatically set by **`symmetry`**:

0	no split of l-like charge
1	p-z, (p-x, p-y) e.g.:hcp
2	e-g, t-2g of d-electrons e.g.:cubic
3	d-z2, (d-xy,d-x2y2), (d-xz,dyz) e.g.:hcp
4	combining option 1 and 3 e.g.:hcp
5	all d symmetries separate
6	all p symmetries separate
8	combining option 5 and 6
-2	d-z2, d-x2y2, d-xy, (d-xz,d-yz)
88	split lm like charges (for **`telnes`**)
99	calculate cross-terms (for **`telnes`**)

>>>: line 5 must now be repeated MULT-1 times for the other positions of each equivalent atom according to the Wyckoff position in the "International Tables of Crystallography".

line 7: format (A10,5X,I5,5X,F10.8,5X,F10.5,5X,F5.2)
name of atom, NPT, R0, RMT, Z

name of atom — Use the chemical symbol. Positions 3-10 for further labeling of nonequivalent atoms (use a number in position 3)

NPT	number of radial mesh points (381 gives a good mesh for LDA calculations, but for GGA twice as many points are recommended; *always use an **odd** number of mesh points!*) the radial mesh is given on a logarithmic scale: $r(n) = R_0 * e^{[(n-1)*DX]}$
R0	first radial mesh point (typically between 0.0005 and 0.00005, smaller for heavy elements, bigger for light ones; a struct-file generated by w2web will have proper R0 values.)
RMT	atomic sphere radius (muffin-tin radius), can easily be estimated after running **nn** (see 6.1) and are set automatically with **setrmt_lapw** see 5.2.6). The following guidelines will be given here: Choose spheres as large as possible as this will save MUCH computer time. But: Use identical radii within a series of calculations (i.e. when you want to compare total energies) — therefore consider first how close the atoms may possibly come later on (volume or geometry optimization); do NOT make the spheres too different (even when the geometry would permit it), instead use the largest spheres for f-electron atoms, 10-20 % smaller ones for d-elements and again 10-20 % smaller for sp-elements; H is a special case, you may choose it much smaller (e.g. 0.6 and 1.2 for H and C) and systems containing H need a much smaller RKMAX value (3-5) in **case.in1**.
Z	atomic number

line 8-10: format (20X,3F10.7)

ROTLOC	local rotation matrix (always in an orthogonal coordinate system). Transforms the global coordinate system (of the unit cell) into the local at the given atomic site as required by point group symmetry (see in the INPUT-Section 7.5.3 of LAPW2). SYMMETRY calculates the point group symmetry and determines ROTLOC automatically. Note, that a proper ROTLOC is required, if the LM values generated by SYMMETRY are used. A more detailed description with several examples is given in the appendix A and sec. 10.3

>>>: lines 5 thru 10 must be repeated for each inequivalent atom
line 11: format (I4)

nsym	number of symmetry operations of space group (see International Tables of Crystallography 64) If nsym is set to zero, the symmetry operations will be generated automatically by SYMMETRY.

line 12-14: format (3I2,F10.7)
matrix, tau (as listed in the International Tables of Crystallography 64)

matrix	matrix representation of (space group) symmetry operation
tau	non-primitive translation vector

line 15: format (I8)
index of symmetry operation specified above
>>>: lines 12 thru 15 must be repeated for all other symmetry operations
(the complete list is contained in sample inputs)

4.4 The "history" file case.scf

During the self-consistent field (SCF) cycle the essential data are appended to the file **case.scf** in order to generate a summary of previous iterations. For an easier retrieval of certain quantities the essential lines are labeled with **:LABEL:**, which can be used to monitor these quantities during self-consistency as explained below. The most important **:LABELs** are

:ENE	total energy (Ry)
:DIS	charge distance between last 2 iterations ($\int \|\rho_n - \rho_{n-1}\| dr$). Good convergence criterium.
:FER	Fermi energy
:FORxx	force on atom xx in mRy/bohr (in the local (for each atom) carthesian coordinate system)
:FGLxx	force on atom xx in mRy/bohr (in the global coordinate system of the unit cell (in the same way as the atomic positions are specified))
:DTOxx	total difference charge density for atom xx between last 2 iterations
:CTOxx	total charge in sphere xx (mixed after MIXER)
:NTOxx	total charge in sphere xx (new (not mixed) from LAPW2+LCORE)
:QTLxx	partial charges in sphere xx
:EPLxx	l-like partial charges and "mean energies" in lower (semicore) energy window for atom xx. Used as energy parameters in case.in1 for next iteration
:EPHxx	l-like partial charges and "mean energies" in higher (valence) energy window for atom xx. Used as energy parameters in **`case.in1`** for next iteration
:EFGxx	Electric field gradient (EFG) V_{zz} for atom xx
:ETAxx	Asymmetry parameter of EFG for atom xx
:RTOxx	Density for atom xx at the nucleus (first radial mesh point)
:VZERO	Gives the total, Coulomb and xc-potential at z=0 and z=0.5 (meaningfull only for slab calculations)

To check to which type of calculation a scf file corresponds use:

:POT	Exchange-correlation potential used in this calculation
:LAT	Lattice parameters in this calculation
:VOL	Volume of the unit cell
:POSxx	Atomic positions for atom xx (as in **`case.struct`**)
:RKM	Actual matrix size and resulting RKmax
:NEC	normalization check of electronic charge densities. If a significant amount of electrons is missing, one might have core states, whose charge density is not completely confined within the respective atomic sphere. In such a case the corresponding states should be treated as band states (using LOs).

For spin-polarized calculations:

:MMTOT	Total spin magnetic moment/cell
:MMIxx	Spin magnetic moment of atom xx. Note, that this value depends on RMT.
:CUPxx	spin-up charge (mixed) in sphere xx
:CDNxx	spin-dn charge (mixed) in sphere xx
:NUPxx	spin-up charge (new, from lapw2+lcore) in sphere xx
:NDNxx	spin-dn charge (new, from lapw2+lcore) in sphere xx
:ORBxx	Orbital magnetic moment of atom xx (needs SO calculations and LAPWDM).
:HFFxx	Hyperfine field of atom xx (in kGauss).

One can monitor the energy eigenvalues (listed for the first k-point only), the Fermi-energy or the total energy. Often the electronic charges per atom reflect the convergence. Charge transfer between the various atomic spheres is a typical process during the SCF cycles: large oscillations should be avoided by using a smaller mixing parameter; monotonic changes in one direction suggest a larger mixing parameter.

In spin-polarized calculations the magnetic moment per atomic site is an additional crucial quantity which could be used as convergence criterion.

If a system has electric field gradients and one is interested in that quantity, one should monitor the EFGs, because these are very sensitive quantities.

It is best to monitor several quantities, because often one quantity is converged, while another still changes from iteration to iteration. The script **`run_lapw`** has three different convergence criteria built in, namely the total energy, the atomic forces and the charge distance (see 5.1.2, 5.1.3).

We recommend the use of UNIX commands like :

grep :ENE case.scf or use "Analysis" from **w2web**

for monitoring such quantities.

You may define an **alias** for this (see sec. 11.2), and a csh-script **grepline_lapw** is also available to get a quantity from several scf-files simultaneously (sec. 5.2.16 and 5.3).

4.5 Flow of programs

The **WIEN2k** package consists of several independent programs which are linked via C-SHELL SCRIPTS described below.

The flow and usage of the different programs is illustrated in the following diagram (Fig. 4.2):

The initialization consists of running a series of small auxiliary programs, which generates the inputs for the main programs. One starts in the respective **case/** subdirectory and defines the structure in **case.struct** (see 4.3). The initialization can be invoked by the script **init_lapw** (see sec. 3.7 and 5.1.2), and consists of running:

NN a program which lists the nearest neighbor distances up to a specified limit (defined by a distance factor f) and thus helps to determine the atomic sphere radii. In addition it is a very usefull additional check of your **case.struct** file (equivalency of atoms)
SGROUP determines the spacegroup of the structure defined in your **case.struct** file.
SYMMETRY generates from a raw case.struct file the space group symmetry operations, determines the point group of the individual atomic sites, generates the LM expansion for the lattice harmonics and determines the local rotation matrices.
LSTART generates free atomic densities and determines how the different orbitals are treated in the band structure calculations (i.e. as core or band states, with or without local orbitals,...).
KGEN generates a k-mesh in the irreducible part of the BZ.
DSTART generates a starting density for the scf cycle by a superposition of atomic densities generated in LSTART.

Then a self-consistency cycle is initiated and repeated until convergence criteria are met (see 3.8 and 5.1.3). This cycle can be invoked with a script **run_lapw**, and consists of the following steps:

LAPW0 (POTENTIAL) generates potential from density
LAPW1 (BANDS) calculates valence bands (eigenvalues and eigenvectors)
LAPW2 (RHO) computes valence densities from eigenvectors
LCORE computes core states and densities
MIXER mixes input and output densities

4.5.1 Core, semi-core and valence states

In many cases it is desirable to distinguish three types of electronic states, namely core, semi-core and valence states. For example titanium has core ($1s$, $2s$, $2p$), semi-core ($3s$, $3p$) and valence ($3d$, $4s$, $4p$) states. In our definition **core states** are only those whose charge is entirely confined inside the corresponding atomic sphere. They are deep in energy, e.g., more than 7-10 Ry below the Fermi energy. **Semi-core states** lie high enough in energy (between about 1 and 7 Ry below the Fermi energy), so that their charge is no longer completely confined inside the atomic sphere, but has a few percent outside the sphere. **Valence states** are energetically the highest (occupied) states and always have a significant amount of charge outside the spheres.

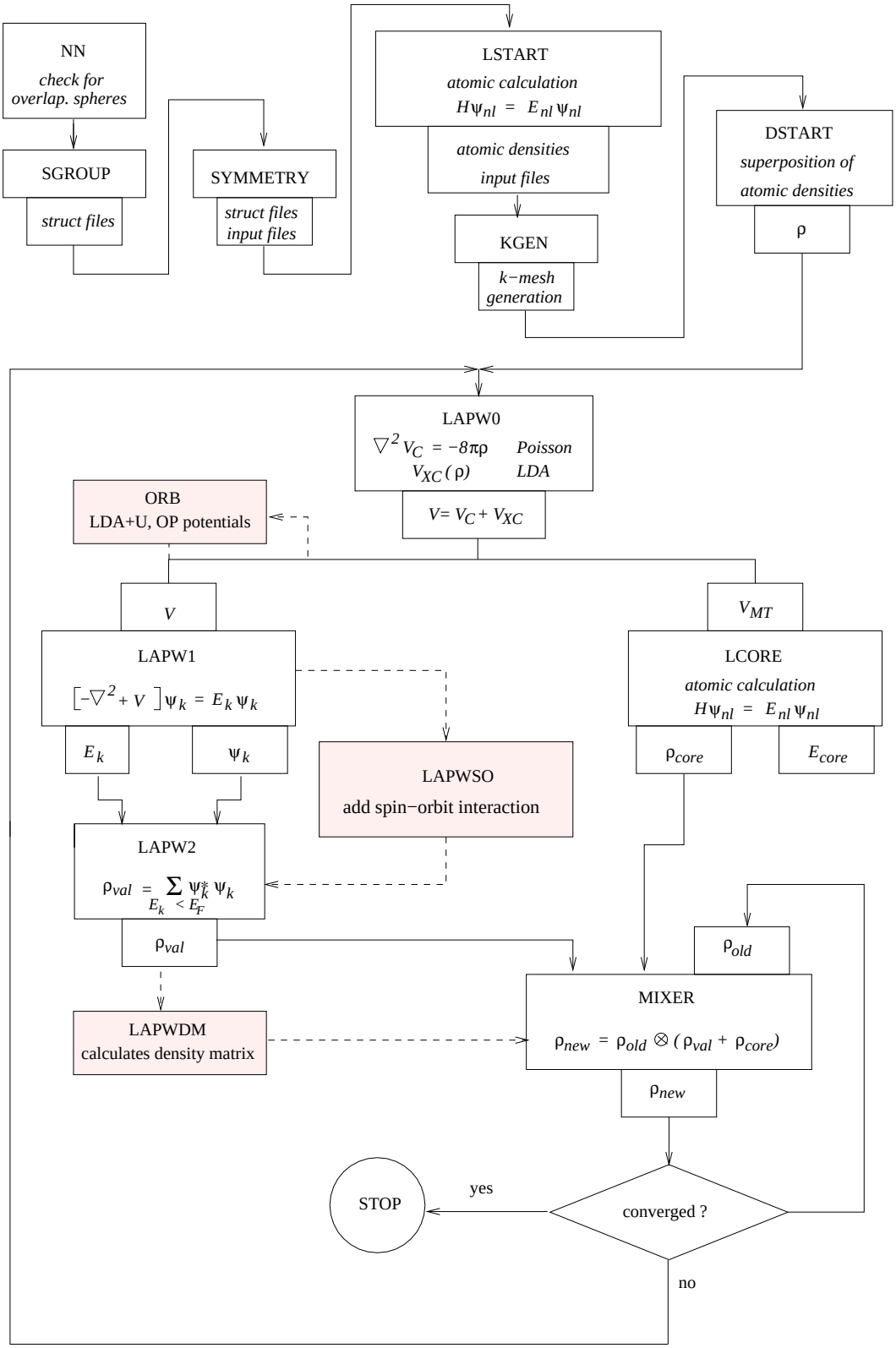

Figure 4.2: Program flow in **WIEN2k**

The energy cut-off specified in **lstart** during **init_lapw** (usually -6.0 Ry) defines the separation into core- and band-states (the latter contain both, semicore and valence). If a system has atoms with semi-core states, then the best way to treat them is with "local orbitals", an extension of the usual LAPW basis. An input for such a basis set will be generated automatically. (Additional LOs can also be used for valence states which have a strong variation of their radial wavefunctions with energy (e.g. d states in TM compounds) to improve the quality of the basis set, i.e. to go beyond the simple linearization).

4.5.2 Spin-polarized calculation

For magnetic systems spin-polarized calculations can be performed. In such a case some steps are done for spin-up and spin-down electrons separately and the script **runsp_lapw** consists of the following steps:

LAPW0 (POTENTIAL) generates potential from density
LAPW1 -up (BANDS) calculates valence bands for spin-up electrons
LAPW1 -dn (BANDS) calculates valence bands for spin-down electrons
LAPW2 -up (RHO) computes valence densities for spin-up electrons
LAPW2 -dn (RHO) computes valence densities for spin-down electrons
LCORE -up computes core states and densities for spin-up electrons
LCORE -dn computes core states and densities for spin-down electrons
MIXER mixes input and output densities

The use of spin-polarized calculations is illustrated for fcc Ni (section 10.2), one of the test cases provided in the **WIEN2k** package.

4.5.3 Fixed-spin-moment (FSM) calculations

Using the script **runfsm_lapw -m XX** it is possible to constrain the total spin magnetic moment per unit cell to a fixed value XX and thus force a particular ferromagnetic solution (which may not correspond to the equillibrium). This is particularly useful for systems with several metastable (non-) magnetic solutions, where conventional spin-polarized calculation would not converge or the solution may depend on the starting density. Additional SO-interaction is not supported.

Please note, that once **runfsm_lapw** has finished, only **case.vectordn** is ok, but **case.vectorup** is NOT the proper up-spin vector and MUST NOT be used for the calculations of QTLs (and DOS). It must be regenerated by **x lapw1 -up** (see also the comments for iterative diagonalization in section 5.2.18).

4.5.4 Antiferromagnetic (AFM) calculations

Several considerations are necessary, when you want to perform an AFM calculation. Please have also a look into $WIENROOT/SRC_afminput/afminput_test.

- You must construct a unit cell which allows for the desired AF ordering. For example for bcc Cr you must select a "P" lattice and specify both atoms, Cr1 at (0,0,0) and Cr2 at (.5,.5,.5), corresponding to a CsCl structure. Note, that it is important to label the two Cr atoms with "Cr1" and "Cr2", since only then the symmetry programs can detect that those atoms should be different (although they have the same Z). *If sgroup has interchanged some axis, try to undo these changes, since afminput may not properly find the correct symmetry operations in such a case.*

- When you generate **case.inst** you must specify the correct magnetic order and flip the spin of the AF atoms (i.e. invert the spin up and dn occupation numbers). In addition you should set a zero moment (identical spin up and dn occupations) for all "non-magnetic" atoms. This can be done conveniently using **instgen_lapw -ask** or during "initialization" using **w2web**.
- Now you can run either a "normal" spinpolarized initialization (without AFM option) and **runsp_lapw** or:
- Create a struct file of the non-magnetic (or ferro-magnetic) supergroup (run init_lapw up to lstart). Name it **case.struct_supergroup**. (For example for bcc Cr, this would be a struct file with the ordinary cubic lattice parameters, "B" type lattice and just one Cr at (0,0,0).)
- Run **init_lapw**. At the end AFMINPUT creates an input file for the program CLMCOPY. Depending on the presence of **case.struct_supergroup** and the specific symmetry it may/may not ask you to supply a symmetry operation/nonprimitive translation (see Sect. 9.5 .
- Run **runafm_lapw**. This script calls LAPW1 and LAPW2 only for spin-up but the corresponding spin-dn density is created by CLMCOPY according to the rules defined during initialization. This reduces the required cpu time by a factor of 2 (and in addition the scf cycle is much more stable).
- It is highly recommended that you save your work (**save_lapw**) and check the results by continuing with a regular **runsp_lapw**. If nothing changes (E-tot and other properties), then you are ok, otherwise make sure the scf calculation is well converged (-cc 0.0001 or better). Eventually the system may not want to be antiferromagnetic (but for instance it is ferrimagnetic!).

runafm_lapw saves you more than a factor of 2 in in computer time, since only spin-up is calculated and in addition the scf-convergence may be MUCH faster. It works also with LDA+U (**case.dmatup/dn** are also copied), but does NOT work with Hybrid-DFT nor spin-orbit coupling, since this requires the presence of both vector files in the LAPWSO step.

4.5.5 Spin-orbit interaction

You can add spin-orbit interaction in LAPWSO (called directly after LAPW1) using a second-variational method with the scalar-relativistic orbitals (from LAPW1) as basis. The number of eigenvalues will double since SO couples spin-up and dn states, so they are no longer separable. In addition, LOs with a "$p_{1/2}$" radial basis can be added. (Kunes et al. 2001)

To assist with the generation of the necessary input files and possible changes in symmetry, a script **initso_lapw** exists. For non-spinpolarized cases nothing particular must be taken into account and SO can be easily applied by running **run_lapw -so**. It will automatically use the complex version of LAPW2.

However, for spin-polarized cases, the SO interaction may change (lower) the symmetry depending on how you choose the direction of magnetization and care must be taken to get a proper setup. **initso_lapw** together with **symmetso** generates the proper symmetry.

Just a few hints what can happen:

- Suppose you have a cubic system and put the magnetization along [001]. This will create a tetragonal symmetry (and you can temporarely tell this to the initialization programs by changing the respective lattice parameter c to a tetragonal system).
- If you put the magnetization along [111], this creates most likely a rhombohedral (or hexagonal) symmetry. (Try to visualize this for a fcc lattice, XCRYSDEN is very usefull for this purpose).

- Symmetry operations can be classified into operations which invert the magnetization,others which leave it unchanged and some which do some arbitrary rotation. The program **symmetso** (part of **initso_lapw**) sorts these operations in the proper way.
- If you don't have inversion symmetry in the original structure, you must not "add inversion" in KGEN.

The recommended way to include SO in the calculations is to run a regular scf calculation first, save the results, initialize SO and run another scf cycle including SO:

- run[sp]_lapw
- save_lapw case_nrel
- initso_lapw
- run[sp]_lapw -so

For spin-polarized systems you may want to add the "-dm" switch to calculate also the orbital magnetic moment.

4.5.6 Orbital potentials

In **WIEN2k**it is possible to go beyond standard LDA (GGA) and include orbital dependent potentials in methods like LDA+U or the "Orbital-Polarization", which are very usefull for strongly correlated systems.

To use these features you need to create input-files for LAPWDM and ORB (**case.indm, case.inorb**). You may copy a template from **SRC_templates**, but must modify it according to your needs. In particular you must select for which atoms and which orbitals (usually d-Orbitals of late transition metal atoms or f-orbitals for 4f/5f atoms) you want to add such a potential and also choose the proper U and J values for them. Once this is done, you can include this using the **-orb** switch. The density matrix (**case.dmatup/dn**) will be calculated after **lapw2** in **lapwdm**, it will be mixed in **mixer** (consistently with the "regular" charge density) and the orbital dependend potentials will be calculated on **orb** (after **lapw0**). Note, you must run spin-polarized in order to use orbital potentials.

- runsp_lapw -orb [-so]

If you want to force a non-magnetic solution you can constrain the spin-polarization to zero using **runsp_c_lapw**.

Without SO, **case.vorbup/dn** will be considered in LAPW1(c). With SO, it will be applied in LAPWSO (and allows coupling of nondiagonal spin-terms).

4.5.7 Exact-exchange and Hybrid functionals for correlated electrons

In **WIEN2k**it is also possible to go beyond standard LDA (GGA) and include on-site exact-exchange (Hartree-Fock), which is very usefull for strongly correlated systems. The exact-exchange/hybrid methods are implemented only inside the atomic spheres, therefore it is recommended to us them only for localized electrons (see Tran et al. 2006 for details). They will NOT improve gaps in sp-semiconductors.

Examples of implemented functionals include:

- **LDA-Hartree-Fock**
  Functional 5 in case.in0. mode = EECE and fraction = 1 in case.ineece.

$$E_{xc}^{\text{LDA-HF}}[\rho] = E_{xc}^{\text{LDA}}[\rho] + E_{x}^{\text{HF}}[\Psi_{\text{corr}}] - E_{xc}^{\text{LDA}}[\rho_{\text{corr}}]$$

- **LDA-Fock-α**
  Functional 5 in case.in0. mode = HYBR and fraction = α in case.ineece.

$$E_{xc}^{\text{LDA-Fock-}\alpha}[\rho] = E_{xc}^{\text{LDA}}[\rho] + \alpha \left(E_x^{\text{HF}}[\Psi_{\text{corr}}] - E_x^{\text{LDA}}[\rho_{\text{corr}}] \right)$$

- **PBE-Fock-α**
  Functional 13 in case.in0. mode = HYBR and fraction = α in case.ineece.

$$E_{xc}^{\text{PBE-Fock-}\alpha}[\rho] = E_{xc}^{\text{PBE}}[\rho] + \alpha \left(E_x^{\text{HF}}[\Psi_{\text{corr}}] - E_x^{\text{PBE}}[\rho_{\text{corr}}] \right)$$

  The PBE0 functional corresponds to $\alpha = 0.25$.
- **PBEsol-Fock-α**
  Functional 19 in case.in0. mode = HYBR and fraction = α in case.ineece.

$$E_{xc}^{\text{PBEsol-Fock-}\alpha}[\rho] = E_{xc}^{\text{PBEsol}}[\rho] + \alpha \left(E_x^{\text{HF}}[\Psi_{\text{corr}}] - E_x^{\text{PBEsol}}[\rho_{\text{corr}}] \right)$$

- **WC-Fock-α**
  Functional 11 in case.in0. mode = HYBR and fraction = α in case.ineece.

$$E_{xc}^{\text{WC-Fock-}\alpha}[\rho] = E_{xc}^{\text{WC}}[\rho] + \alpha \left(E_x^{\text{HF}}[\Psi_{\text{corr}}] - E_x^{\text{WC}}[\rho_{\text{corr}}] \right)$$

- **TPSS-H-Fock-α**
  Functional 27 in case.in0. mode = HYBR and fraction = α in case.ineece.

$$E_{xc}^{\text{TPSS-H-Fock-}\alpha}[\rho] = E_{xc}^{\text{TPSS}}[\rho] + \alpha \left(E_x^{\text{HF}}[\Psi_{\text{corr}}] - E_x^{\text{TPSS}}[\rho_{\text{corr}}] \right)$$

  It is similar to PBE0, but uses the meta-GGA TPSS.
- **B3PW91**
  Functional 18 in case.in0. mode = HYBR and fraction = 0.2 in case.ineece.

$$\begin{aligned} E_{xc}^{\text{B3PW91}}[\rho] &= E_{xc}^{\text{LDA}}[\rho] + 0.2 \left(E_x^{\text{HF}}[\Psi_{\text{corr}}] - E_x^{\text{LDA}}[\rho_{\text{corr}}] \right) \\ &\quad + 0.72 \left(E_x^{\text{B88}}[\rho] - E_x^{\text{LDA}}[\rho] \right) \\ &\quad + 0.81 \left(E_c^{\text{PW91}}[\rho] - E_c^{\text{LDA}}[\rho] \right) \end{aligned}$$

In addition to the input files which are necessary for an usual LDA or GGA calculation, the input file **`case.ineece`** is necessary to start a calculation. You may copy a template from **`SRC_templates`**, but must modify it according to your needs. In particular you must select for which atoms and which orbitals (usually d-Orbitals of late transition metal atoms or f-orbitals for 4f/5f atoms) you want to add such a potential and which type of functional you want to use.

A sample input for calculations with exact exchange is given below.

```
------------------ top of file: case.ineece -----------
-9.0  2       emin, natorb
1  1  2       1st atom index, nlorb, lorb
2  1  2       2nd atom index, nlorb, lorb
HYBR          HYBR / EECE mode
0.25          fraction of exact exchange
------------------ bottom of file ---------------------
```

Interpretive comments on this file are as follows:
line 1: free format
emin, natom

emin	lower energy cutoff, to be selected so that the energy of correlated states is larger than emin
natorb	number of atoms for which the exact exchange is calculated

line 2: free format
iatom(i), nlorb(i), (lorb(li,i), li=1,nlorb(i))

iatom	index of atom in struct file
nlorb	number of orbital moments for which exact exchange shall be calculated
lorb	orbital numbers (repeated nlorb-times)

2^{nd} **line repeated natorb-times**
line 3: free format
mode

HYBR	means that LDA/GGA exchange will be replaced by exact exchange
EECE	means that LDA/GGA exchange-correlation will be replaced by exact exchange

line 4: free format

alpha	This is the fraction of Hartree-Fock exchange (between 0 and 1)

As with LDA+U, hybrid functionals can be used only for spin-polarized calculations (**runsp_lapw** with the switch **-eece**). **runsp_lapw** will internally call **runeece_lapw**, which will create all necessary additional input files (it requires a **case.in0** file including the optional IFFT line as generated by **init_lapw**): **case.indm (case.indmc), case.inorb, case.in0eece, case.in2eece (case.in2ceece)** and once this is done, calculates in a series of **lapw2/lapwdm/lapw0/orb** calculations the corresponding orbital dependend potentials.

- runsp_lapw -eece [-so]

4.5.8 modified Becke-Johnson potential (mBJ) for band gaps

The modified Becke-Johnson exchange potential + LDA-correlation (Tran and Blaha 2009) allows the calculation of band gaps with an accuracy similar to very expensive GW calculations. It is a local approximation to an atomic "exact-exchange"-potential and a screening term. This is just a XC-potential, not a XC-energy functional, thus E_{xc} is taken from LSDA and the forces cannot be used with this option.

We recommend the following steps to perform such calculations:

- run a regular initialization and scf cycle using LDA or PBE,
- create **case.inm_vresp** (**cp $WIENROOT/SRC_templates/case.inm_vresp case.inm_vresp**.
- edit **case.in0** and set "R2V" option (instead of "NR2V").
- run one more scf-cycle (use **run_lapw -NI -i 1**) to generate the required **case.vresp*** files.
- "save" the LDA (PBE) calculation.
- edit **case.in0** and change the functional to option indxc=28.

- **cp case.in0 case.in0_grr** and change indxc in **case.in0_grr** to 50. This option will calculate the average of $\nabla\rho/\rho$ over the unit cell. (The presence of **case.in0_grr** will be detected during the scf-cycle and **lapw0** will be called twice, first with the input file **case.in0_grr**, then with **case.in0**.)
- edit **case.inm** and change to PRATT mixing.
- run another scf cycle.

As mentioned above, you should use PRATT mixing since in most cases MSEC1 mixing will lead to convergence problems of the scf cycle (V also depends on the kinetic energy density and this is not mixed in **mixer**). PRATT mixing can be slow, or can start to oszillate and diverge.Thus, first using a smaller mixing factor (eg. 0.2), later increasing it to about 0.50 to make sure that you do not stop at false convergence.

The TB-mBJ potential uses an average of $\nabla\rho/\rho$ over the unit cell. This does not make sense for surfaces or molecules. In such cases, run a similar bulk structure first, then cp **case_bulk.grr** to **case.grr** and remove **case.in0_grr**. This runs mBJ with a fixed value of "c".

5 Shell scripts for running programs

Contents

5.1 Job control (c-shell scripts)

In order to run **WIEN2k** several c-shell scripts are provided which link the individual programs to specific tasks.

All available (user-callable) commands have the ending **_lapw** so you can easily get a list of all commands using

```
ls $WIENROOT/*_lapw
```

in the directory of the **WIEN2k** executables. (*Note: all of the more important commands have a link to a short name omitting "_lapw".*) All these commands have at least one option, **-h**, which will print a small help indicating purpose and usage of this command.

5.1.1 Main execution script (x_lapw)

The main **WIEN2k**script, **x_lapw** or **x**, executes a single **WIEN2k**program. First it creates the corresponding **program.def**-file, where the connection between Fortran I/O-units and filenames are defined. One can modify its functionality with several switches, modifying file definitions in case of spin-polarized or complex calculations or tailoring special behaviour. All options are listed with the help switch

x -h or **x_lapw -h**

With some of the options the corresponding input files may be changed temporarely, but are set back to the original state upon completion.

```
USAGE:  x PROGRAMNAME [flags]

PURPOSE:runs WIEN executables:  afminput,aim,arrows,broadening,cif2struct,
      clmaddsub,clmcopy,clminter,dipan,dstart,eosfit,eosfit6,filtvec,init_xspec,
      hex2rhomb,irrep,joint,join_vectorfiles, kgen,kram,lapw0,lapw1,lapw2,lapw3,
      lapw5,lapw7,lapwdm,lapwso,lcore,lorentz,lstart,mini,mixer,nn,pairhess,,
      plane,qtloptic,optimize,orb,rhomb_in5,sgroup,spaghetti,struct_afm_check,
      sumpara,supercell,symmetry,symmetso,telnes3,tetra,txspec,xspec, dmftproj

FLAGS:
-f FILEHEAD ->  FILEHEAD for path of struct & input-files
-t/-T ->        suppress output of running time
-h/-H ->        help
-d    ->        create only the def-file
-up   ->        runs up-spin
-dn   ->        runs dn-spin
-du   ->        runs up/dn-crossterm
-sc   ->        runs semicore calculation
-c    ->        complex calculation (no inversion symmetry present)
-p    ->        run lapw1/2/so in parallel (needs .machines file)
-orb  ->        runs lapw1 with LDA+U/OP or B-ext correction
-it   ->        runs lapw1 with iterative diagonalization
-noHinv   ->    runs lapw1 with iterative diag. without  Hinv
-noHinv0  ->    runs lapw1 with iterative diag. writing new Hinv
-nohns->        runs lapw1 without HNS
-nmat_only->    runs lapw1 and yields only the matrixsize
-in1orig  ->    runs lapw2 but does not modify case.in1
-emin X ->      runs lapw2 with EMIN=X (in bin9_blaha.in2)
-all X Y ->     runs lapw2 with ALL and E-window X-Y (in bin9_blaha.in2)
-qtl  ->        calculates QTL in lapw2
-alm  ->        calculates ALM,BLM in lapw2
-almd  ->       calculates ALM,BLM in lapw2 for DMFT (Aichhorn/Georges/Biermann)
-qdmft ->       calculates charges including DMFT (Aichhorn/Georges/Biermann)
-help_files ->  creates case.helpXX files in lapw2
-vresp->        creates case.vrespval (for TAU/meta-GGA) in lapw2
-eece ->        for hybrid-functionals (lapw0,lapw2,mixer,orb,sumpara)
-grr  ->        lapw0 for grad rho/rho (using SRC.in0_grr with ixc=50)
-band ->        for bandstructures: unit 4 to 5 (in1), sets QTL and ROOT (in2)
-fermi->        calculates Fermi energy and weights in lapw2
-efg  ->        calculates lapw2 with EFG switch
-so   ->        runs lapw2 with def-file for spin-orbit calculation
-fbz   ->       runs kgen and generates a full mesh in the BZ
-fft  ->        runs dstart only up to case.in0_std creation
-super->        runs dstart and creates new_super.clmsum (and not case.clmsum)
-sel  ->        use reduced vector file in lapw7
-settol 0.000x -> run sgroup with different tolerance
-sigma->        run lstart with case.inst_sigma (autogenerated) for diff.dens.
-rxes->         run tetra using case.rxes weight file for RXES-spectroscopy.
-rxesw E1 E2->  run tetra and create case.rxes file for RXES for energies E1-E2
-delta->        run arrows program with difference between two structures
-lcore->        runs dstart with SRC.rsplcore (produces SRC.clmsc)
-copy ->        runs pairhess and copies .minpair to .minrestart and .minhess
-telnes  ->     run qtl after generating case.inq based on case.innes
USE: x -h PROGRAMNAME   for valid flags for a specific program
```

Note: To make use of a scratch file system, you may specify such a filesystem in the environment variable ***SCRATCH*** *(it may already have been set by your system administrator). However, you have to make sure that there is enough disk-space in the* ***SCRATCH*** *directory to hold your* ***case.vector**** *and* ***case.help**** *files.*

5.1.2 Job control for initialization (init_lapw)

In order to start a new calculation, one should make a new subdirectory and run all calculations from there. At the beginning one must provide at least one file (see 3), namely **case.struct** (see 4.3) (**case.inst** can be created automatically on the "fly", see 6.4.3), then one runs a series of programs using **init_lapw**. This script is described briefly in chapter 4.5) and in detail in "Getting started" for the example TiC (see chapter 3). You can get help with switch -h. All actions of this script are logged in short in **:log** and in detail in the file **case.dayfile**, which also gives you a "restart" option when problems occurred. In order to run **init_lapw** starting from a specific point on, specify -s PROGRAM.

Ignoring ERRORS and in many cases also WARNINGS during the execution of this script, most likely will lead to errors at a later stage. Neglecting warnings about core-leakage creates **.lcore**, which directs the scf-cycle to peform a superposition of core densities.

init_lapw supports switch **-b**, a "batch" mode (non-interactive) for trivial cases AND experienced users. You can supply various options and specify spin-polarization, XC-potential, RKmax, k-mesh or mixing. See **init_lapw -h** for more details. Changes to **case.struct** by **nn** will be accepted, but by **sgroup** will be neglected. *Please check the terminal output for ERRORS and WARNINGS !!!*

5.1.3 Job control for iteration (run_lapw or runsp_lapw)

In order to perform a complete SCF calculation, several types of scripts are provided with the distribution. For the specific flow of programs see chapter 4.5.

- For non-spinpolarized calculations use: **run_lapw**,
- for spin-polarized calculations use: **runsp_lapw**.
- for antiferromagnetic calculations use: **runafm_lapw**
- for FSM (fixed-spin moment) calculations use: **runfsm_lapw**
- for a spin-polarized setup, where you want to constrain the moment to zero (e.g. for LDA+U calculations) use: **runsp_c_lapw**

Cases with/without inversion symmetry and with/without semicore or core states are handled automatically by these scripts. All activities of these scripts are logged in short in **:log** (appended) and in detail together with convergence information in **case.dayfile** (overwriting the old "dayfile"). You can always get help on its usage by invoking these scripts with the -h flag.

run_lapw -h

```
PROGRAM:        /zeus/lapw/WIEN2k/bin/run_lapw

PURPOSE:        running the nonmagnetic scf-cycle in WIEN
                to be called within the case-subdirectory
                has to be located in WIEN-executable directory

USAGE:          run_lapw [OPTIONS] [FLAGS]

OPTIONS:
-cc LIMIT ->    charge convergence LIMIT (0.0001 e)
-ec LIMIT ->    energy convergence LIMIT (0.0001 Ry)
-fc LIMIT ->    force  convergence LIMIT (1.0 mRy/a.u.)
                default is -ec 0.0001; multiple convergence tests possible
-e PROGRAM ->   exit after PROGRAM ()
-i NUMBER ->    max. NUMBER (40) of iterations
-s PROGRAM ->   start with PROGRAM ()
-r NUMBER ->    restart after NUMBER (99) iterations (rm *.broyd*)
-nohns NUMBER ->do not use HNS for NUMBER iterations
-in1new N ->    create "new" in1 file after N iter (write_in1 using scf2 info)
-ql LIMIT ->    select LIMIT (0.05) as min.charge for E-L setting in new in1
-qdmft NP ->    including DMFT from Aichhorn/Georges/Biermann running on NP proc

FLAGS:
-h/-H ->        help
-I    ->        with initialization of in2-files to "TOT"
-NI   ->        does NOT remove case.broyd*  (default: rm *.broyd* after 60 sec)
-p    ->        run k-points in parallel (needs .machine file [speed:name])
-it   ->        use iterative diagonalizations
-it1  ->        use iterative diag. with recreating H_inv (after basis change)
-it2  ->        use iterative diag. with reinitialization (after basis change)
-noHinv   ->    use iterative diag. without H_inv
-vec2pratt ->   use vec2pratt instead of vec2old for iterative diag.
-so   ->        run SCF including spin-orbit coupling
-renorm->       start with mixer and renormalize density
-in1orig->      if present, use case.in1_orig file; do not modify case.in1
```

```
CONTROL FILES:
.lcore          runs core density superposition producing case.clmsc
.stop           stop after SCF cycle
.fulldiag       force full diagonalization
.noHinv         remove case.storeHinv files
case.inm_vresp  activates calculation of vresp files for meta-GGAs
case.in0_grr    activates a second call of lapw0 (mBJ pot., or E_xc analysis)

ENVIRONMENT VARIBLES:
SCRATCH         directory  where vectors and help files should go
```

Additional flags valid only for magnetic cases (**runsp_lapw**) include:

```
-dm   ->        calculate the density matrix (when -so is set, but -orb is not)
-eece ->        use "exact exchange+hybrid" methods
-orb  ->        use LDA+U, OP or B-ext correction
-orbc ->        use LDA+U correction, but with constant V-matrix
```

Calling **run_lapw** (after **init_lapw**) from the subdirectory **case** will perform up to 40 iterations (or what you specified with switch -i) unless convergence has been reached earlier. You can choose from three convergence criteria, -ec (the total energy convergence is the default and is set to 0.0001 Ry for at least 3 iterations), -fc (magnitude of force convergence for 3 iterations, ONLY if your system has "free" structural parameters!) or -cc (charge convergence, just the last iteration), and any combination can also be specified. Be careful with these criteria, different systems will require quite different limits (e.g. fcc Li can be converged to μRy, a large unit cell with heavy magnetic atoms only to 0.1 mRy). You can stop the scf iterations after the current cycle by generating an empty file **.stop** (use eg. **touch .stop** in the respective case-directory).

The scf-cycle creates **case.broyd*** files which contain the "charge-history". Once run_lapw has finished, you should usually "**save_lapw**" (see below) the results. When you continue with another run_lapw without "**save_lapw**" (because the previous run did not fulfill the convergence criteria or you want to specify a more strict criterium) the "broyden-files" will be deleted unless you specify -NI.

With -e PROGRAM you can run only part of one scf cycle (e.g. run lapw0, lapw1 and lapw2), with -s PROGRAM you can start at an arbitrary point in the scf cycle (e.g. after a previous cycle has crashed and you want to continue after fixing the problem) and continue to self-consistency. Before mixer is invoked, **case.clmsum** is copied to **case.clmsum_old**, and the final "important" files of the scf calculation are **case.clmsum** and **case.scf**.

Invoking

```
run_lapw -I -i 30 -fc 0.5
```

will first set in case.in2 the TOT-switch (if FOR was set) to save cpu time, then run up to 30 scf cycles till the force criterion of 0.5 mRy/a.u. is met (for 3 consecutive iterations). Then the calculation of all terms of the forces is activated (setting FOR in **case.in2**) for a final iteration.

By default the file **case.in1** is updated after **lapw2** and the current Fermi-energy is inserted. This will force **lapw1** to use instead of the default energy parameters (**0.30**) an energy "$E_F - 0.2$". The switch **-in1orig** can be used to keep the present **case.in1** file unmodified (or to copy **case.in1_orig** back after -in1new).

The switch **-in1new N** preserves for N iteration the current **case.in1** file. After the first N iterations **write_in1_lapw** is called and a new **case.in1** file is generated, where the energy parameters are set according to the :EPLxx and :EPHxx values of the last scf iteration and the **-ql** value (see sections 4.4 and 7.3). In this way you may select in some cases better energy-parameters and also additional LOs to improve the linearization may be generated automatically. Note, however, that this option is potentially unsave and dangerous, since it may set energy-parameters of LOs and APW+lo too close (leading to ghostbands) or in cases where you have a "bad" last iteration (or large changes from one scf iteration to the next. The original **case.in1** file is saved in **case.in1_orig** and is used as template for all further scf-cycles.

Parallelization is described in Sec. 5.5.

Iterative diagonalization, which can significantly save computer time (in particular for cases with "few electrons" (like surfaces) and "large matrices (larger than 2000)" a factor 2-5 ! is possible), is described in Sec. 7.3. It needs the **case.vector.old** file from the previous scf-iteration (and this file is created from **case.vector** when the -it switch is set) and an inverse of a previous Hamiltonian (H_0^{-1}) stored in **case.storeHinv**. When you change the Hamiltonian significantly (changing RKmax or local orbitals), reinitialize the iterative diagonalization either by "**touch .fulldiag**" (performs one full diagonalization) or "**touch .noHinv**" (recreates **case.storeHinv** files) or using the **-it1|-it2** switch.

You can save computer time by performing the first scf-cycles without calculating the non-spherical matrix elements in lapw1. This option can be set for **N** iterations with the **-nohns N** switch.

The presence of the file **.lcore** directs the script to superpose the radial core densities using **dstart** and generating **case.clmsc**. It is created automatically during **init_lapw** when charge-leakage warnings are ignored. This option allows to reduce the number of semi-core states, but still keeping a good charge density. Unfortunately, **dstart** is not parallelized and thus can be slow for big cases.

The presence of the file **case.in0_grr** activates a second call of **lapw0**, which is necessary for modified Becke-Johnson potentials (see Section 4.5.8) or E_{xc} analysis.

If you have a previous scf-calculation and changed lattice parameters or positions (volume optimization or internal positions minimization), one could use **-renorm** to renormalize the density prior to the first iteration., but the recommended way is to use **clmextrapol_lapw**.

For magnetic systems which are difficult to converge you can use the script **runfsm_lapw -m M** (see section 4.5.3) for the execution of fixed-spin moment (FSM) calculations.

5.2 Utility scripts

5.2.1 Save a calculation (save_lapw)

After self-consistency has been reached, the script

```
save_lapw head_of_save_filename
```

saves **case.clmsum**, **case.scf**, **case.dmat**, **case.vorb** and **case.struct** under the new name and removes the **case.broyd*** files. Now you are ready to modify structural parameters or input switches and rerun **run_lapw**, or calculate properties like charge densities (lapw5), total and partial DOS (tetra) or energy bandstructures (spaghetti).

For more complicated situations, where many parameters will be changed, we have extended **save_lapw** so that calculations can not only be saved under the **head_of_save_filename** but also a directory can be specified. If you use any of the possible switches (-a, -f, -d, -s) all input files will be saved as well (and can be restored using **restore_lapw**).

Options to **save_lapw** can be seen with

```
save_lapw -h
```

Currently the following options are supported

-h	help
-a	save **all** input files as well
-f	force save_lapw to overwrite previous saves
-d directory	save calculation in directory specified
-s	silent operation (no output)

5.2.2 Restoring a calculation (restore_lapw)

To restore a calculation the script **restore_lapw** can be used. This script restores the **struct**, **clmsum**, **vorb** and **dmat** files as well as all input files. **Note**: The input files will only be restored when **save_lapw -d** was used, i.e. when you have saved a calculation in an extra directory.

After **restore_lapw** you can continue and either run an scf cycle (**run_lapw**) or recreate the scf-potential (**x lapw0**) and the corresponding eigenvectors (**x lapw1**) for further tasks (DOS, electron density,...).

Options to restore_lapw are:

-h	help
-f	force restore_lapw to overwrite previous files
-d directory	restore calculation from directory specified
-s	silent operation (no output)
-t	only test which files would be restored

5.2.3 Remove unnecessary files (clean_lapw)

Once a case has been completed you can clean up the directory with this command. Only the most important files (scf, clmsum, struct, input and some output files) are kept. It is very important to use this command when you have finished a case, since otherwise the large vector and helpXX files will quickly fill up all your disk space.

5.2.4 Migrate a case to/from a remote computer (migrate_lapw)

This script migrates a case to a remote computer (to be called within the case-dir). Needs working ssh/scp without password; local and remote case-dir must have the same name.

Call it within the desired case-dir as:

migrate_lapw [FLAGS OPTIONS] [user@]host:path/case-dir

with the following options:

```
-put            -> transfer of files to a remote host (default)
-get            -> transfer of files from a remote host

-all            -> the complete directory is copied
-start          -> only files to start an scf cycle are copied (default for put)
-end            -> only new files resulting from an scf cycle are copied
                   (default for get)
-save savedir   -> "save_lapw -d save_dir" is issued and only save_dir is copied

FLAGS:
-h              -> help
-clean          -> a clean_lapw is issued before copying
-r              -> files in source directory are removed after copying
-R              -> source directory (and all files) are removed after copying
-s              -> do it silent (in batch mode)
-z              -> gzip files before scp (slow network)
```

5.2.5 Generate case.inst (instgen_lapw)

This script generates **case.inst** from a **case.struct** file. It is used automatically in init_lapw, if **case.inst** is not present. Using some options (see below) it allows to define the spin-state of all/certain atoms. Note: the label "RMT" is necessary in **case.struct**.

```
instgen_lapw [-h -s -up -dn -nm -ask]
   -h:   generate this message
   -s:   silent operation (do not ask)
   -up:  generates spin-up configuration for all atoms (default)
   -dn:  generates spin-dn configuration for all atoms
   -nm:  generates non-magnetic configuration for all atoms
   -ask: asks for each atom which configuration it should generate
```

5.2.6 Set R-MT values in your case.struct file (setrmt_lapw)

This perl-script executes **x nn** and uses its output to determine the atomic sphere radii (obeying recommended ratios for H, sp-, d- and f- elements). It is called automatically within **init_lapw** or you may call it separately using:

setrmt_lapw case [-r X]

where **case** gives the head of the **case.struct** file. You may specify a reduction of the RMTs by X percent in order to allow for structural optimizations. It creates **case.struct_setrmt**.

5.2.7 Create case.int file (for DOS) (configure_int_lapw)

This script creates the input file **case.int** for the program **tetra** and allows to specifiy which partial DOS (atom, l and m) should be calculated. It was provided by Morteza Jamal (m_jamal57@yahoo.com).

You can specify interactively:

```
  total        (for plotting 'Total Dos')
  N            (to select atom N)
    s,p,d,...  (to select a set of PDOS for previously selected atom N)
               use labels as listed in the header of your case.qtl file)
  end          (for exit)
```

There is also a "batch" (non-interactive) mode:

```
configure_int_lapw -b  total 1 tot,d,d-eg,d-t2g  2 tot,s,p  end
```

which will prepare case.int (eg. for the TiC example) with:

```
tic             #Title
 -1.000   0.00250   1.200  0.003     #Emin, DE, Emax, Gauss-Broad
     8                                #Number of DOS
     0 1 total-DOS
     1 1 tot-Ti
     1 4 d-Ti
     1 5 d-eg-Ti
     1 6 d-t2g-Ti
     2 1 tot-C
     2 2 s-C
     2 3 p-C
```

5.2.8 Check for running WIEN jobs (check_lapw)

This script searches for **.running.*** files within the current directory (or the directory specified with "-d full_path_directory") and then performs a **ps** command for these processes. If the specified process has not been found, it removes the corresponding **.running.*** file after confirmation (default) or immediately (when "-f" has been specified).

5.2.9 Cancel (kill) running WIEN jobs (cancel_lapw)

This script searches for **.running.*** files within the current directory (or the directory specified with "-d full_path_directory") and then kills the corresponding process after confirmation (default) or immediately (when "-f" has been specified). It is particular usefull for killing "k-point parallel" jobs.

5.2.10 Extract critical points from a Bader analysis (extractaim_lapw)

This script extracts the critical points (CP) after a Bader analysis (**x aim (-c)**) from **case.outputaim**. It sorts them (according to the density), removes duplicate CPs, converts units into $\AA, e/\AA^3, ...$ and produces **critical_points_ang**.

It is used with: **extractaim_lapw case.outputaim**

5.2.11 scfmonitor_lapw

This program was contributed by:

Hartmut Enkisch
Institute of Physics E1b
University of Dortmund
Dortmund, Germany
enkisch@pop.uni-dortmund.de

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

It produces a plot of some quantities as function of iteration number (a maximum of 6 quantities is possible at once) from the **case.scf** file as specified on the commandline using **analyse_lapw** and GNUPLOT. This plot is updated in regular intervals.

You can call **scfmonitor_lapw** using:

```
scfmonitor_lapw [-h] [-i n] [-f case.scf] [-p] arg1 [arg2 ..
arg6]
```

```
-h              help switch
-i n            show only the last n iterations
-f scf-file     use  "scf-file"  instead of the default  "case.scf"
-p              produces file  "scfmonitor.png"  instead of X-window plot
arg1,...        arguments to monitor (like  ":ENE"  or  ":DIS" , see analyse_lapw )
```

The **scfmonitor** can also be called directly from **w2web** using the "Analyse" tool.

In order to have a reasonable behavior of scfmonitor the GNUPLOT window should stay in background. This can be achieved by putting a line into your **.Xdefaults** file like:

gnuplot*raise: off

*Note: It does not make sense to start **scfmonitor** before the first cycle has finished because no **case.scf** exists at this point.*

5.2.12 analyse_lapw

The script **analyse_lapw** is usually called from **scfmonitor_lapw**. It "greps" from an scf-file the specified arguments and produces **analyse.out**.

analyse_lapw is called using:

```
analyse_lapw [-h] scf-file arg1 [arg2 arg3 arg4 arg5 arg6]
```

```
-h              help switch
scf-file        "scf-file"  to analyse (there's no default  "case.scf" !)
arg1,...        arguments to analyse:
                atom independend:   :ENE :DIS :FER :MMT
                atom iii dependend: :CTOiii :CUPiii :CDNiii :NTOiii :NUPiii :NDNiii
                                    :DTOiii :DUPiii :DDNiii :RTOiii :EFGiii :HFFiii
                                    :MMIiii
                vector quantities:  :FORiii[x/y/z] :POSiii[x/y/z] :FGLiii[x/y/z]
                             where     magnitude      z              z   is the defaul
```

For vector quantities like :FGLiii or :POSiii (usefull with **case.scf_mini**) one can specify the respective coordinate by adding x/y/z to the corresponding labels.

5.2.13 Check parallel execution (testpara_lapw)

testpara_lapw is a small script which helps you to determine an optimal selection for the file **.machines** for parallel calculations (see sec. 5.5).

5.2.14 Check parallel execution of lapw1 (testpara1_lapw)

testpara1_lapw is a small script which determines how far the execution of **lapw1para** has proceeded.

5.2.15 Check parallel execution of lapw2 (testpara2_lapw)

testpara2_lapw is a small script which determines how far the execution of **lapw2para** has proceeded.

5.2.16 grepline_lapw

Using

grepline_lapw :label 'filename*.scf' lines_for_tail or

grepline :label 'filename*.scf' lines_for_tail

you can get a list of a quantity "**:label**" (e.g. :ENE for the total energy) from several scf files at once.

5.2.17 initso_lapw

initso_lapw helps you to initialize the calculations for spin-orbit coupling. It helps together with **make_inso_lapw** (contributed by Morteza Jamal, m_jamal57@yahoo.com) to create/modify all required input files (**case.inso, case.in1, case.in2c**). In a spinpolarized case SO may reduce symmetry or equivalent atoms may become non-equivalent, and the script calls **symmetso** and will help you to find proper symmetries and setup the respective input files. It is called using

initso_lapw or

initso

and you should *carefully* follow the instructions and explanations of the script and the explanations for **case.inso** given in section 7.4.

5.2.18 vec2old_lapw

vec2old_lapw moves **case.vector** files to **case.vector.old**. Usually called automatically just before **lapw1** when the iterative diagonalization (**run_lapw -it**) is specified. It also works for the k-parallel case including local $SCRATCH directories (add -p as first argument, uses hosts from **.processes** and requires commensurate k-point/number_of_processors) and spin-polarization (-up/-dn switches).

For **runfsm_lapw** the sequence had to be changed and the switches **-updn** or **-dnup** forces vec2old to COPY **case.vectorup** to**case.vectordn** (and vice versa). In the **runfsm_lapw** case the corresponding **case.vector*.old** files are generated just AFTER **lapw2/lapwdm** and not BEFORE **lapw1**. Thus after **runfsm_lapw** has finished, the corresponding spin-up/dn vectors are **case.vector*.old** and NOT **case.vector***.

The switches **-p -local** will copy $SCRATCH/case.vector* to case.vector*. It will be done automatically when you run **x lapw2 -p -qtl**.

An alternative script **vec2pratt_lapw** was provided by L.D.Marks (l-marks@northwestern.edu) which together with **SRC_vecpratt** mixes the last two vectors (Pratt mixing) to generate **case.vector.old**. It is activatd using the **-vec2pratt** switch in **run_lapw**.

5.2.19 clmextrapol_lapw

clmextrapol_lapw extrapolates the charge density (**case.clmsum/up/dn**) from old to new positions (or from old to new lattice parameters). It takes the density from the old positions (copied into **old.clmsum**) and subtracts an atomic superposition density (**new_super.clmsum**) fom the old positions and adds an atomic superposition density fom the new ones (generated by **dstart**).

If **new_super.clmsum** (generated automatically by **init_lapw**) is not present, it will be generated and for the next geometry step an extrapolation will take place.

It is usually called from "**min_lapw** " after a geometry step has finished and a new struct file has been generated.

It can significantly reduce the number of scf-cycles for the new geometry step.

5.3 Structure optimization

5.3.1 Lattice parameters (Volume, c/a, lattice parameters)

Package optimize

The auxilliary program **optimize** (**x optimize**) generates from an existing **case.struct** (or **case_initial.struct**, which is generated at the first call of **optimize**) a series of struct files with various volumes (or c/a ratios, or other modified parameters) (depending on your input):

```
 [1]  VARY VOLUME with CONSTANT RATIO A:B:C
 [2]  VARY C/A RATIO with CONSTANT VOLUME (tetr and hex lattices)
 [3]  VARY C/A RATIO with CONSTANT VOLUME and B/A (orthorh lattice)
 [4]  VARY B/A RATIO with CONSTANT VOLUME and C/A (orthorh lattice)
 [5]  VARY A and C (2D-case) (tetragonal or hexagonal lattice)
 [6]  VARY A, B and C (3D-case) (orthorhombic lattice)
 [7]  VARY A, B, C and Gamma (4D-case) (monoclinic lattice)
 [8]  VARY C/A RATIO and VOLUME (2D-case) (tetr and hex lattices)
```

It also produces a shell-script **optimize.job** which looks similar to:

```
#!/bin/csh -f
 foreach i ( \
        tic_vol_-10.0  \
        tic_vol__-5.0  \
        tic_vol___0.0  \
        tic_vol___5.0  \
        tic_vol__10.0  \
 )
     cp  $i.struct tic.struct
 #    cp  $i.clmsum tic.clmsum
 #    x dstart
 #    run_lapw -ec 0.0001 -in1new 3  -renorm
     run_lapw -ec 0.0001
     set stat = $status
     if ($stat) then
        echo "ERROR status in" $i
        exit 1
     endif
     save_lapw  ${i}
 #    save_lapw  -f -d XXX $i
 end
```

You may modify this script according to your needs: use **runsp_lapw** or even **min_lapw**, or specify different convergence parameters; modify the **save_lapw** command and change the save-name or save into a directory to separate e.g. "gga" and "lda" results. Eventually you may activate the line " **cp $i.clmsum case.clmsum**" to use a previously saved clmsum file, e.g. from a calculation with smaller RKmax, ... and deactivate the "clmextrapol_lapw" lines, but usually the latter is so efficient that this is no longer recommended.

*Note: You must have a **case.clmsum** file (either from **init_lapw** or from a previous scf calculation) in order to run **optimize.job**.*

After execution of this script you should have a series of scf-files with energies corresponding to the modified parameters, which should allow you to find the corresponding equillibrium parameters. For the volume optimization an analysis tool is available, other tools are under development).

Using the script **grepline** (or the "*Analysis ⇩ Analyze multiple SCF-files*" menu of **w2web**) you get a summary of the total energy vs. volume (c/a). The file **case.analysis** can be used in **eplot_lapw** to find the minimum total energy and the equilibrium volume (c/a). Supported equation of states include the EOS2, Murnaghan and Birch-Murnaghan EOS.

```
grepline :ENE '*.scf' 1 > case.analysis
grepline :VOL '*.scf' 1 >> case.analysis
```

Using such strategies also higher-dimensional optimizations (e.g. c/a ratio and volume) are possible in combination with the -d option of **save_lapw**.

For optimization of more degrees of freedom (2-4 lattice parameters), you can use the corresponding option and for analysis of the data the script **parabolfit_lapw** together with the program **eosfit6**. It performs a non-linear least squares fit, using a parabolic fit-function in your variables and get an analytic description of your energy surface. Please note, this is only a harmonic fit (no odd or higher terms) and the description may not be very good if your parameter range is large and/or the function is quite anharmonic, or you suffer from numerical noise.

For the determination of elastic constants see the description of ELAST in sec 8.14.

Package 2Doptimize

This program was contributed by:

⇩ Morteza Jamal
email: m_jamal57@yahoo.com
Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This package performs a convenient 2D structure optimization (Volume and c/a, i.e. tetragonal or hexagonal spacegroups). After initialization of a **case**, one generates a set of structures and a job-file **2Doptimize.job** using the command

```
set2D_lapw
```

This calls **setup2D** and you have to specify the "order of the fit" (stored in **.fordfitcoa**) and the changes in volume and c/a. The resulting **2Doptimize.job** script should be adapted (eg. use min_lapw instead of run_lapw; insert switches,...) and executed. Finally

```
ana2D_lapw
```

can be executed and will analyze the results. It uses a set of **case.Vconst*** files (produced by **2Doptimize.job** and stored also in subdirectory **Vconst**) and the **numbvcoa** file. You can modify the order of the fits in **.fordfitcoa** and rerun **ana2D_lapw** to check the sensitivity of the results with the fitting. Note: Fits of high order (and few "data points") may lead to artificial results due to unphysical oszillations of the fit.

You can see results for
- energy vs. c/a for each volume,
- energy vs. volume (with optimized c/a) and
- c/a vs. volume.

Optionally you can specify more cases by rerunning **set2D_lapw**. Specify also your **``old'' volume and c/a points again** (or leave them out on purpose in case they were very bad (eg. very far from the minimum). The old results will then be taken automatically into account without recalculation (unless you modify **2Doptimize.job**, see the comments at the top of this file). Thus a "good" strategy is to use only 3x3 points (order of fit = 3) and in a second step you add points where they are needed.

When you want to rerun such an optimization with different parameters (RKmax, k-mesh, XC-potentials) modify the top of **2Doptimize.job** and set *answscf=no* and a new *savename* (eg. "_pbe_rk8_1000k").

5.3.2 Minimization of internal parameters (min_lapw)

Most of the more complicated structures have free internal structural parameters, which can either be taken from experiment or optimized using the calculated **forces on the nuclei**.

Starting with WIEN2k_11.1 there are two possibilities to determine the equilibrium position of all individual atoms automatically (obeying the symmetry constraints of a certain space group). One can use either

- the shell script **min_lapw**, together with the program **mini**, which will run a scf-cycle, update the positions using the calculated forces and restarts a new scf cycle. This continues until forces drop below a certain value;
- or use the normal scf-scripts **run_lapw** with a special input in **case.inm** such that the charge density and the positions are simultaneously optimized during the scf-cycle.

At present we recommend the second (new) option, although there are cases where this scheme can be slower or may even fail to converge.

A typical sequence of commands for an optimization of the internal positions would look like:

- Generate struct file
- **init_lapw**
- **run_lapw -fc 1** [another runXX script or additional options are of course also possible] (this may take some time)
- Inspect the scf file whether you have significant forces (usually at least .gt. 5 mRy/bohr), otherwise you are more or less at the optimal positions (An experienced user may omit the run_lapw step and proceed directly from init_lapw to the next step)

Now you have to decide which method to use:

- **min_lapw [options]** (this may take some time)
  - it will generate a default **case.inM** (if not present) by:
    - executing "x pairhess -copy ; cp case.inM_st case.inM " (i.e. it sets up the PORT minimization option and calculates an approximate starting Hessian).
    - when -nohess is specified, it will generate case.inM from SRC_templates with the NEW1 option (not recommended).
  - Without -NI switch min_lapw performs an initialization first:
    - removes "histories" (case.broyd*, case.tmpM) if present;
    - copies **.min_hess** to **.minrestart** (if present from previous min_lapw or x pairhess).

- or edit **case.inm** and put MSR1a (or MSEC1a) as "mixing method". Then continue with **run_lapw -fc 0.5 -ec 0.0001 -cc 0.001 [-it]**. It will run **x pairhess** (unless **case.inM** is already present) and then run (several hundreds) scf-cycles, simultaneously updating positions and charge densities. Once the forces seem to be smaller than the limit defined in **case.inM** it will switch to "mixing method" MSR1 and finalize the scf-cycle with fixed positions. Because of this, the final forces may not be as small as desired and eventually you have to restart this step using MSR1a again.

When using the second method we recommend you read carefully $WIENROOT/SRC_mixer/Mixer_README_4.1.pdf. Overall the method is very good for semiconductors (or well behaved metals), and allows "tricks" like small k-mesh or small RKMax at the beginning of the minimization and using higher accuracy only towards the end.

The following text refers (mainly) to the first method using **min_lapw**:

When **case.scf** is not present, an scf-cycle will be performed first, otherwise the corresponding forces are extracted into **case.finM** and the program **mini** generates a new **case.struct** with modified atomic positions. The previous step is saved under **case_1/2/3...**. Then a new scf-cycle is executed and this loop continues until convergence (default: forces below 2mRy/bohr) is reached.
The last iteration of each geometry step is appended to **case.scf_mini**, so that this file contains the complete history of the minimization and can be used to monitor the progress (grep :ENE *mini; or :FORxxx ...).

By default (unless switch **-noex** is specified), min will call the script **clmextrapol_lapw** after the first geometry step and try to extrapolate the charge density to the new positions. This procedure usually significantly reduces the number of scf-cycles and is thus highly recommended.

mini requires an input file **case.inM** (see Sec. 8.15) which is created automatically and MUST NOT be changed while **min_lapw** is running (except the force tolerance, which terminates the optimization).

We recommend the PORT minimization method, a reverse-communication trust-region Quasi-Newton method from the Port library, which seems to be stable, efficient and does not depend too much on the users input (DELTAs, see below with NEWT). The PORT option also uses/produces a file **.min_hess**, which contains the (approximate) Hessian matrix (lower-triangle Cholesky factor) If you restart a minimization with different k-points, RMT, RKmax, ... or do a similar calculation (eg. for a different volume, ...) it will be copied to **.minrestart** (unless -nohess is specified), so that you start with a reasonable approximation for the Hessian. The program **pairhess**, which calculates the first Hessian, also prints out the average Hessian eigenvalue for the symmetric, symmetry-preserving modes in mRyd/au2 as well as the minimum and maximum, and also the vibration frequencies. A list of these is given at the end of **case.pairhess**. Note that these are not all possible modes, but only the symmetry preserving ones. Therefore if you have prior information about the vibrations of the system you can adjust the rescaling term so the average vibration frequency is about correct. (see the description of pairhess in 9.2). (In addition there is a program **eigenhess**, which will analyze the Hessian after the minimization has been completed. It also prints vibrational frequencies and may give you hints about dynamical instability of your system. Some more description is given in **$WIENROOT/SRC_pairhess/README** and at the top of the output file **case.outputeig**.

When using PORT you may also want to check its progress using

```
grep :LABEL  case.outputM
```

where :LABEL is :ENE (should decrease), :GRAD (should also go down, but could sometimes also go up for some time as long as the energy still decreases), :MIN (provides a condensed summary of the progress), :WARN may indicate a problem), :DD (provides information about the step sizes and mode used). Some general explanations are:

1) The algorithm takes steps along what it considers are good directions (using some internal logic), provided that these steps are smaller than what is called the trust-region radius. After a good step (e.g. large energy decrease) it expands the trust-region; after a bad one it reduces it. Sometimes it will try too large a step then have to reduce it, so the energy does not always go down. You can see this by using ":DD" and ":MIN" .
2) A grep on :MIN gives a condensed progress output, in which the most significant terms are E (energy in some rescaled units), RELDF (last energy reduction), PRELDF (what the algorithm predicted for the step), RELDX (RMS change in positions in Angstroms) and NPRELDF (predicted change in next cycle). Near the solution RELDF and RELDX should both become small. However, sometimes you can have soft modes in your structure in which case RELDX will take a long time before it becomes small.
3) A warning that the step was reduced due to overlapping spheres if it happens only once (or twice) is not important; the algorithm tested too large a step. However, if it occurs many times it may indicate that the RMT's are too big.
4) A warning "CURVATURE CONDITION FAILED" indicates that you are still some distance from the minimum, and the Hessian is changing a lot. If you see many of these, it may be that the forces and energy are not consistent.

Sometimes PORT gets "stuck" (often because of inconsistencies of energy and forces due to insufficient scf convergence or a very non-harmonic potential energy surface). A good alternative is NEW1, which is a "sophisticated" steepest-descent method with optimized step size. It can be very efficient in certain cases, but can also be rather slow when the potential energy surface is rather flat in one, but steep in another direction (eg. a weakly bound molecule on a surface, but constraining the sensitive parameters, like the bond distance of the molecule, may help).

Another alternative is NEWT, where one must set proper "DELTAs" and a "FRICTION" for each atom. Unfortunately, these DELTAs determine crucially how the minimization performs. Too small values lead to many (unnecessary) "geometry steps", while too large DELTAs can even lead to divergence (and finally to a crash). Thus you **MUST** control how the minimization performs. We recommend the following sequence after 2-3 geometry steps:

```
grep :ENE *mini
:ENE  : ********** TOTAL ENERGY IN Ry =        -2994.809124
:ENE  : ********** TOTAL ENERGY IN Ry =        -2994.813852
:ENE  : ********** TOTAL ENERGY IN Ry =        -2994.818538
```

Good, since the total energy is decreasing.

```
grep :FGL001 *mini
:FGL001:  1.ATOM                          0.000          0.000         18.219
:FGL001:  1.ATOM                          0.000          0.000         12.375
:FGL001:  1.ATOM                          0.000          0.000          7.876
```

Good, since the force (only a force along z is present here) is decreasing reasonably fast towards zero. You must check this for every atom in your structure.

When you detect oszillations or too small changes of the forces during geometry optimization, you will have to decrease/increase the DELTAs in **case.inM** and **rm case.tmpM**. (NOTE: You must not continue with modified DELTAs but keeping **case.tmpM**.) Alternatively, stop the minimization (**touch .minstop** and wait until the last step has finished), change **case.inM** and restart.

You can get help on its usage with:

min -h or **min_lapw -h**

```
PROGRAM:        min

USAGE:          min [OPTIONS]

OPTIONS:
-j JOB ->       job-file JOB (default: run_lapw -I -fc 1. -i 40 )
-noex ->        does not extrapolate the density for next geometry step
-p ->           adds -p (parallel) switch to run_lapw
-it ->          adds -it (iterative diag.) switch to run_lapw
-it1 ->         adds -it1 (it.diag. with recreating H_inv) switch to $job
-it2 ->         adds -it2 (it.diag. with reinitialization) switch to $job
-noHinv ->      adds -it -noHinv (it.diag. without H_inv) switch to $job
-sp ->          uses runsp_lapw instead of run_lapw
-nohess ->      removes .minrestart (initial Hessian) from previous minimization
-m ->           extract force-input and execute mini (without JOB) and exit
-mo ->          like -m but without copying of case.tmpM1 to case.tmpM
-h/-H ->        help
-NI ->          without initialization of minimization (eg. continue after a crash)
-i NUMBER ->    max. NUMBER (50) of structure changes
-s NUMBER ->    save_lapw after NUMBER of structure changes

CONTROL FILES:
.minstop        stop after next structure change
```

For instance for a spin-polarized case, which converges more difficultly, you would use:

min -j ``runsp_lapw -I -fc 1.0 -i 60''

5.4 Phonon calculations

Calculations of phonons is based on a program PHONON by K.Parlinski, which runs under MS-Windows and must be ordered separately (see http://wolf.ifj.edu.pl/phonon/). Alternatively you may also try the package PHONOPY by Atsushi Togo (see http://www.wien2k.at//reg_user/unsupported/).

You would define the structure of your compound in PHONON together with a supercell of sufficient size (e.g. 64 atoms). PHONON will then generate a list of necessary displacements of the individual atoms. The resulting file **case.d45** must be transfered to UNIX. Here you would run WIEN2k-scf calculations for all displacements and collect the resulting forces, which will be transfered back to PHONON (**case.dat** and/or **case.dsy**). With these force information PHONON calculates phonon at arbitrary q-vectors together with several thermodynamic properties.

5.4.1 init_phonon_lapw

init_phonon_lapw uses **case.d45** from PHONON and creates subdirectories **case_XX** and **case_XX.struct** files for all required displacements. It allows you to define globally RMT values for the different atoms and
- initializes every case individually (batch option of init_lapw is now supported) or
- initializes every second case (useful for pos. and neg. displacements, which have the same symmetry and thus only one initialization is necessary), or
- initializes only the first case and copies the files from the first case to all others. This is most convenient in low symmetry cases with P1 symmetry for all cases and thus just one init_lapw needs to be executed (while for higher symmetry a separate initialization is required (but computational effort is reduced).

Please use mainly **nn** to reduce equivalent atoms. **sgroup** might change the unitcell and than the collection of forces into the original supercell is not possible (or quite difficult).

A script **run_phonon** has been created. Modify it according to your needs (parallelization,....) and run all cases to selfconsistency.

Note that good force convergence is essential (at least 0.1 mRy/bohr) and if your structure has free parameters, either very good equillibrium positions must have been found before, or even better, use both, positive and negative displacements to average out any resulting error from non-equillibrium positions.

5.4.2 analyse_phonon_lapw

analyse_phonon_lapw uses the resulting scf files and generates the "Hellmann-Feynman"-file required by PHONON. When you have positive and negative displacements an automatic averaging will be performed. The resulting **case.dat** and **case.dsy** filse should be transfered back to MS-Windows and imported into PHONON.

5.5 Running programs in parallel mode

This section describes two methods for running **WIEN2k** on parallel computers.

One method, parallelizing k-points over processors, utilizes c-shell scripts, NFS-file system and passwordless login ((public/private keys). This method works with all standard flavors of Unix without any special requirements. The parallelization is very efficient even on heterogeneous computing environments, e. g. on heterogeneous clusters of workstations, but also on dedicated parallel computers and does NOT need large network bandwidth.

The other parallelization method, which comes new with version **WIEN2k_07.3**, is based on fine grained methods, MPI and SCALAPACK. It is especially useful for larger systems, if the required memory size is no longer available on a single computer or when more processors than k-points are available. It requires a fast network (at least Gb-Ethernet, better Myrinet or Infiniband) or a shared memory machine. Although not as efficient as the simple k-point parallelization, the current mpi-version has been enhanced a lot and shows very good scaling with the number of processors for most parts. In any case, the number of processors and the size of the problem (number of atoms, matrixsize due to the plane wave basis) must be compatible and typically [NMAT / sqrt(processors)] .gt. 2000 should hold.

The k-point parallelization can use a dynamic load balancing scheme and is therefore usable also on heterogeneous computing environments like networks of workstations or PCs, even if interactive users contribute to the processors' work load.

If your case is large enough, but you still have to use a few k-points, a combination of both parallelization methods is possible (always use k-point parallelism if you have more than 1 k-point).

5.5.1 k-Point Parallelization

Parts of the code are executed in parallel, namely **LAPW1, LAPWSO, LAPW2, LAPWDM,** and **OPTIC**. These are the numerically intensive parts of most calculations.

Parallelization is achieved on the k-point level by distributing subsets of the k-mesh to different processors and subsequent summation of the results. The implemented strategy can be used both on a multiprocessor architecture and on a heterogeneous (even multiplatform) network.

To make use of the k-point parallelization, make sure that your system meets the following requirements:

NFS: All files for the calculation must be accessible under the same name and path. Therefore you should set up your NFS mounts in such a way, that on all machines the path names are the same.

Remote login: rlogin or **ssh** to all machines must be possible **without** specifying a password. Therefore you must either edit your **.rhosts** file to include all machines you intend to use (not necessary for a shared memory machine), or correctly specify public/private keys for **ssh**. This can be done by running "**ssh-keygen -t rsa**" and copying the **id_rsa.pub** key into **~/.ssh/authorized_keys** at the remote sites.
The command for launching a remote shell is platform dependent, and usually can be '**ssh**', '**rsh**' or '**remsh**'. It should be specified during installation when **siteconfig_lapw** is executed (see chapter 11).

5.5.2 MPI parallelization

Fine grained MPI parallel versions are available for the programs **lapw0, lapw1,** and **lapw2**. This parallelization method is based on parallelization libraries, including MPI, ScaLapack, PBlas and FFTW_2.1.5 (lapw0). The required libraries are not included with **WIEN2k**. On parallel computers, however, they are usually installed. Otherwise, free versions of these libraries are available[1].

The parallelization affects the naming scheme of the executable programs: the fine grained parallel versions of **lapw0/1/2** are called **lapw0_mpi**, **lapw1[c]_mpi**, and **lapw2[c]_mpi**. These programs are executed by calls to the local execution environments, as in the sequential case, by the scripts **x, lapw0para, lapw1para,** and **lapw2para**. On most computers this is done by calling **mpirun** and should also be configured using **siteconfig_lapw**.

5.5.3 How to use WIEN2k as a parallel program

To start the calculation in parallel, a switch must be set and an input file has to be prepared by the user.

- The switch **-p** switches on the parallelization in the scripts **x** and **run_lapw**.
- In addition to this switch the file **.machines** has to be present in the current working directory. In this file the machine names on which the parallel processes should be launched, and their respective relative speeds must be specified.

If the **.machines** file does not exist, or if the **-p** switch is omitted, the serial versions of the programs are executed.

Generation of all necessary files, starting of the processes and summation of the results is done by the appropriate scripts **lapw1para**, **lapwsopara**,**lapwdmpara** and **lapw2para** (when using **-p**), and parallel programs **lapw0_mpi**, **lapw1_mpi**, and **lapw2_mpi** (when using fine grained parallelization has been selected in the **.machines** file).

5.5.4 The .machines file

The following **.machines** file describes a simple example. We assume to have 5 computers, (alpha, ... epsilon), where epsilon has 4, and delta and gamma 2 cpus. In addition, gamma, delta and epsilon are 3 times faster than alpha and beta.:

```
# This is a valid .machines file
#
granularity:1
1:alpha
1:beta
```

[1] http://www-unix.mcs.anl.gov/mpi/mpich, http://www.netlib.org/scalapack, http://www.fftw.org/

```
3:gamma:2 delta
3:delta:1 epsilon:4
residue:delta:2
lapw0:gamma:2 delta:2 epsilon:4
```

To each set of processors, defined by a single line in this file, a certain number of k-points is assigned, which are computed in parallel. In each line the weight (relative speed) and computers are specified in the following form:

```
weight:machine_name1:number1 machine_name2:number2 ...
```

where **weight** is an integer (e.g. a three times more powerful machine should have a three times higher weight). The name of the computer is **machine_name[1/2/...]**, and the number of processors to be used on these computers are **number[1/2/...]**. If there is only one processor on a given computer, the **:1** may be omitted. Empty lines are skipped, comment lines start with #.

Assuming there are 8 k-points to be distributed in the above example, they are distributed as follows. The computers **alpha** and **beta** get 1 each. Two processors of computer **gamma** and one processor of computer **delta** cooperate in a fine grained parallelization on the solution of 3 k-points, and one processor of computer **delta** plus four processors of computer **epsilon** cooperate on the solution of 3 k-points. If there were additional k-points, they would be calculated by the first processor (or set of processors) becoming available. With higher numbers of k-points, this method ensures dynamic load balancing. If a processor is busy doing other (e. g., interactive) work, the overall calculation will not stall, but most of its work will be done by other processors (or sets of processors using MPI). This is, however, not an implementation for fail safety: if a process does not terminate (e. g., due to shutdown of a computer) the calculation will never terminate. It is up to the user to handle with such hardware failures by modifying the **.machines** file and restarting the calculation at the appropriate point.

During the run of **lapw1para** the file **.processes** is generated. This file is used by **lapw2para** to determine which **case.vector*** to read.

By default **lapw1para** will generate approximately 3 vector-files per processor, if enough k-points are available for distribution. The factor 3 is called "granularity" and allows for some load balancing in heterogeneous environments. If you can be sure that load balancing is not an issue (eg. because you use a queuing-system and can be sure that you will get 100% of the cpus for your jobs) it is recommended to set

```
granularity:1
```

for best performance.

On shared memory machines it is advisable to add a "residue machine" to calculate the surplus (residual) k-points (given by the expression MOD($klist, \sum_j newweight_j$) and rely on the operating system's load balancing scheme. Such a "residue machine" is specified as

```
residue:machine_name:number
```

in the **.machines** file.

Alternatively, it is also possible to distribute the remaining k-points one-by-one (and not in one junk) over all processors. The option

```
extrafine:1
```

can be set in the **.machines** file.

When using "**iterative diagonalization**" or the **$SCRATCH** variable (set to a local directory), the k-point distribution must be "fixed". This means, the ratio (k-points / processors) must be integer (sloppy called "commensurate" at other places in the UG) and granularity:1 should be set.

The line

```
lapw0:gamma:2 delta:2 epsilon:4
```

defines the computers used for running **lapw0_mpi**. In this example the 6 processors of the computers **gamma**, **delta**, and **epsilon** run **lapw0_mpi** in parallel.

If fine grained parallelization is used, each set of processors defined in the **.machines** file is converted to a single file **.machine[1/2/...]**, which is used in a call to **mpirun** (or another parallel execution environment).

When using a queuing system (like PBS, LoadLeveler or SUN-Gridengine) one can only request the NUMBER of processors, but does not know on which nodes the job will run. Thus a "static" **.machines** file is not possible. On can write a simple shell script, which will generate this file on the fly once the job has been started and the nodes are assigned to this job. Examples can be found at our web-site "http://www.wien2k.at/reg_users/faq".

5.5.5 How the list of k-points is split

In the setup of the k-point parallel version of LAPW1 the list of k-points in **case.klist** (*Note, that the k-list from* ***case.in1*** *cannot be used for parallel calculations*) is split into subsets according to the weights specified in the **.machines** file:

$$newweight_i = \left\lfloor \frac{weight_i * klist}{granularity * \sum_j weight_j} \right\rfloor$$

where $newweight_i$ is the number of k-points to be calculated on processor i. $newweight_i$ is always set to a value greater equal one.

A loop over all i processors is repeated until all k-points have been processed.

Speedup in a parallel program is intrinsically dependent on the serial or parallel parts of the code according to Amdahl's law:

$$speedup = \frac{1}{(1-P) + \frac{P}{N}}$$

whereas N is the number of processors and P the percentage of code executed in parallel.

In **WIEN2k** usually only a small part of time is spent in the programs **lapw0, lcore** and **mixer** which is very small (negligible) in comparison to the times spent in **lapw1** and **lapw2**. The time for waiting until all parallel **lapw1** and **lapw2** processes have finished is important too. For a good performance it is therefore necessary to have a good load balancing by estimating properly the speed and *availability* of the machines used. We encourage the use of **testpara_lapw** or "*Utils.* *testpara*" from **w2web** to check the k-point distribution over the machines *before* actually running the programs in parallel.

While running **lapw1** and **lapw2** in parallel mode, the scripts **testpara1_lapw** (see 5.2.14) and **testpara2_lapw** (see 5.2.15) can be used to monitor the succession of parallel execution.

5.5.6 Flow chart of the parallel scripts

To see how files are handled by the scripts **lapw1para** and **lapw2para** refer to figures 5.1 and 5.2. After the **lapw2** calculations are completed the densities and the informations from the **case.scf2_x** files are summarized by **sumpara**.

*Note: parallel **lapw2** and **sumpara** take two command line arguments, namely the **case.def** file but also a **number_of_processor** indicator.*

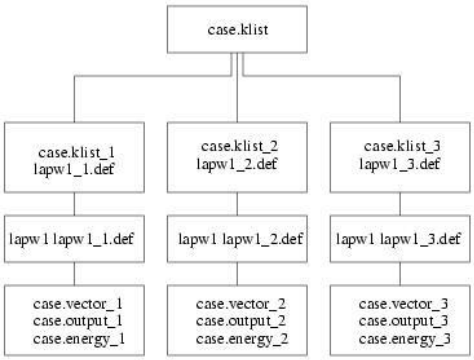

Figure 5.1: Flow chart of **lapw1para**

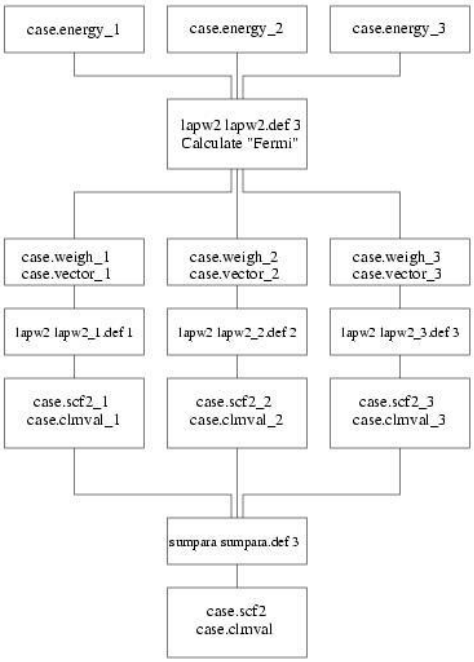

Figure 5.2: Flow chart of **lapw2para**

5.5.7 On the fine grained parallelization

The following parallel programs use different parallelization strategies:

lapw0_mpi is parallelized over the number of atoms. This method leads to good scalability as long as there are more atoms than processors. For very many processors, however, the speedup is limited, which is usually not at all critical, since the overall computing time of **lapw0_mpi** is quite small.

lapw1_mpi uses a two-dimensional processor setup to distribute the Hamilton and overlap matrices. For higher numbers of processors two-dimensional communication patterns are clearly preferable to one-dimensional communication patterns.
Let us assume, for example, 64 processors. In a given processing step, one of these processors has to communicate with the other 63 processors if a one-dimensional setup was chosen. In the case of a two-dimensional processor setup it is usually sufficient to communicate with the processors of the same processor row (7) or the same processor column (7), i. e. with 14 processors.
In general the processor array $P \times Q$ is chosen as follows: $P = \lfloor\sqrt{\text{number of processors}}\rfloor$, $Q = \left\lfloor \frac{\text{number of processors}}{P} \right\rfloor$. Because of SCALAPACK, often $P \times P$ arrays (i.e. 4, 9, 16,... processors) give best performance and of course it is not recommended to use eg. 17 processors.

lapw2_mpi is parallelized in two main parts: (i) The density inside the spheres is parallelized over atoms, and (ii) the fast Fourier transforms are done in parallel.
In addition the density calculation for each atom can be further parallelized by distributing the eigenvector on a certain subset of processors (ususlly 2-4). This is not so efficient, but most usefull if the memory requirement is too big otherwise. You set it in **.machines** using

```
lapw2_vector_split:2
```

If more than one k-point is distributed at once to **lapw1_mpi** or **lapw2_mpi**, these will be treated consecutively.

Depending on the parallel computer system and the problem size, speedups will vary to some extend. Matrix setup in lapw1 should scale nearly perfect, while diagonalization (using SCALAPACK) will not. Usually, "iterative" scales better than "full" diagonalization and is preferred for large scale computations. Scalability over atoms will be very good if processor and atom numbers are compatible. Running the fine grained parallelization over a 100 Mbit/s Ethernet network is not recommended, even for large problem sizes.

5.6 Getting on-line help

- As mentioned before, all **WIEN2k** csh-shell scripts have a **"help"-switch -h**, which gives a brief summary of all options for the respective script.
- To obtain online help on input-parameters, program description, ... use

  ```
  help_lapw
  ```

  which opens the pdf-version of the users guide (using **acroread** or what is defined in $PDFREADER). You can search for a specific keyword using "^f keyword". This procedure substitutes an "Index" and should make it possible to find a specific information without reading through the complete users guide.
- In addition there is a html-version of the UG and its starting page is:
  $WIENROOT/SRC_usersguide_html/usersguide.html
- When using the user interface **w2web**, you have access to the html and pdf-version (the latter requires an X-windows environment) of the usersguide.
- At our webserver $http://www.wien2k.at/reg\_user$ we put informations for the registered user:
  - A "FAQ" page with answers to some common problems.

- Update information: When you think the program has an error, please check wether newer versions are available, which might have fixed the problem you encounter.
- A mailing list:

  Please check the "digest"! In many cases your questions may have been answered before.

  Locate your problem: If a calculation crashes, please locate the problem. Check the content of files like **`case.dayfile`**, **`*.error`**, **`case.scf`**, **`case.scfX`**, **`case.outputX`** where X specifies the program which crashed.

  Posting questions: Please provide enough information so that somebody can help you. A question like: "My calculation crashed. Please help me!" will most likely not be answered.

5.7 Interface scripts

We have included a few "interface scripts" into the current **WIEN2k** distribution, to simplify the previewing of results. In order to use these scripts the public domain program "**gnuplot**" has to be installed on your system.

5.7.1 eplot_lapw

The script **`eplot_lapw`** plots total energy vs. volume or total energy vs. c/a-ratio using the file **`case.analysis`**. The latter should have been created with **`grepline`** (using :VOL and :ENE labels) or the "*Analysis ⇩ Analyze multiple SCF-files*" menu of **w2web** and the file names must be generated (or compatible) with "optimize.job".

For a description of how to use the script for batch like execution call the script using

```
eplot_lapw -h
```

5.7.2 parabolfit_lapw

The script **`parabolfit_lapw`** is an interface for a harmonic fitting of E vs. 2-4-dim lattice parameters by a non-linear least squares fit (eosfit6) using PORT routines. Once you have several scf calculations at different lattice parameters (usually generated with **`optimize.job`**) it generates the required **`case.ene`** and **`case.latparam`** from your scf files. Using

```
parabolfit_lapw [ -t 2/3/4 ] [ -f FILEHEAD ] [ -scf '*xxx*.scf' ]
```

you can optionally specify the dimensionality of the fit or the specific scf-filenames.

5.7.3 dosplot_lapw

The script **`dosplot_lapw`** plots total or partial Density of States depending on the input used by **`case.int`** and the interactive input. A more advanced plotting interface is provided by **`dosplot2_lapw`**, see below.

For a description of how to use the script for batch like execution call the script using

```
dosplot_lapw -h
```

5.7.4 dosplot2_lapw

The script **dosplot2_lapw** plots total or partial Density of States depending on the input used by **case.int** and the interactive input. It can plot up to 4 DOS curves into one plot, and simultaneously plot spin-up/dn DOS.

It was provided by Morteza Jamal (m_jamal57@yahoo.com).

For a description of how to use the script for batch like execution call the script using

```
dosplot2_lapw -h
```

5.7.5 Curve_lapw

The script **Curve_lapw** plots x,y data from a file specified interactively. It asks for additional interactive input. It can plot up to 4 curves into one plot and is a simple gnuplot interface.

It was provided by Morteza Jamal (m_jamal57@yahoo.com).

5.7.6 specplot_lapw

specplot_lapw provides an interface for plotting X-ray spectra from the output of the **xspec** or **txspec** program.

For a description of how to use the script for batch like execution call the script using

```
specplot_lapw -h
```

5.7.7 rhoplot_lapw

The script **rhoplot_lapw** produces a surface plot of the electron density from the file **case.rho** created by lapw5.

*Note: To use this script you must have installed the C-program **reformat** supplied in **SRC_reformat**.*

5.7.8 opticplot_lapw

The script **opticplot_lapw** produces XY plots from the output files of the optics package using the **case.joint**, **case.epsilon**, **case.eloss**, **case.sumrules** or **case.sigmak**. For a description of how to use the script for batch like execution call the script using

```
opticplot_lapw -h
```

5.7.9 addjoint-updn_lapw

The script **addjoint-updn_lapw** adds the files **case.jointup** and **case.jointdn** together and produces **case.joint**. It uses internally the program **add_columns**. It should be called for spin-polarized optics calculations after **x joint -up** and **x joint -dn**, because the Kramers-Kronig transformation to the real part of the dielectric function (ϵ_1) is not a simple additive quantity concerning the spin (see Ambrosch-Draxl 06). The KK transformation should then be done non-spinpolarized (**x kram**) resulting in files: **case.epsilon**, **case.eloss**, **case.sumrules** or **case.sigmak**.

6 Programs for the initialization

Contents

In sections (6.1-6.6) we describe the initial utility programs. These programs are used to set up a calculation.

6.1 NN (nearest neighbor distances)

This program uses the **case.struct** file (see 4.3) in which the atomic positions in the unit cell are specified, calculates the nearest neighbor distances of all atoms, and checks that the corresponding atomic spheres (radii) are not overlapping. If an overlap occurs, an error message is shown on the screen. In addition, the next nearest-neighbor distances up to f times the nearest-neighbor distance (f must be specified interactively) are written to an output file named **case.outputnn**. For negative f values only the distances of non-equivalent atoms are printed. , but equivalent ones are not listed again. Optionally one can specify also a "dlimit" parameter, which helps nn to find equivalent atoms in case of "inaccuarate" structural data.

It is highly recommended in most cases that you change your sphere sizes and do NOT use the default of 2.0. An increase from 2.0 to 2.1 may already result in drastically reduced computing time. More recommendations are given in chapter 4.3.

nn also checks if equivalent atoms are specified correctly in **case.struct**. At the bottom of **case.outputnn** the coordination shell-structure is listed and from that a comparison with the input is made verifying that equivalent atoms really have equivalent environments. If this is not the case, an ERROR will be printed and a new structure file **case.struct_nn** is generated. You have to recheck your input and then decide whether you want to accept the new structure file, or reject it (because the equivalency may just be an artefact due to a special choice of lattice parameters). It also may be that you have made a simple input error. If you want to force two atoms of the same kind (e.g. 2 Fe atoms) to be nonequivalent (e.g. because you want to do an antiferromagnetic calculation), label the atoms as "Fe1" and "Fe2" in **case.struct**.

Thus this program helps to generate proper **struct**-files especially in the case of artificial unit cells, e.g. a supercell simulating an impurity or a surface.

It also prints the "bond-valences" (see also the comments in $WIENROOT/SRC_nn/BVA).

6.1.1 Execution

The program **nn** is executed by invoking the command:

nn nn.def or **x nn**

6.2 SGROUP

This program was contributed by:

Bogdan Yanchitsky and Andrei Timoshevskii
Institute of Magnetism, Kiev, Ukraine
email: yan@imag.kiev.ua and tim@ukron.kiev.ua
Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

It was published in Yanchitsky and Timoshevskii 2001, and is written in C.

This program uses information from **case.struct** (lattice type, lattice constants, atomic positions) and determines the spacegroup as well as all pointgroups of non-equivalent sites. It uses the nuclear charges Z or the "label" in the 3rd place of the atomic name (Si1, Si2) to distinguish different atoms uniquely. It is able to find possible smaller unit cells, shift the origin of the cell and can even produce a new struct file **case.struct_sgroup** based on your input **case.struct** with proper lattice types and equivalency. It is thus most usefull in particular for "handmade" structures.

For more information see also the README in SRC_sgroup.

6.2.1 Execution

The program sgroup is executed by invoking the command:

sgroup -wi case.struct [-wo case.struct_sgroup] case.outputsgen
or **x sgroup**

6.3 SYMMETRY

This program uses information from **case.struct** (lattice type, atomic positions). If NSYM was set to zero it generates the space group symmetry operations and writes them to **case.struct_st** to complete this file. Otherwise (NSYM $>$ 0) it compares the generated symmetry operations with the already present ones. If they disagree a warning is given in the output. In addition the point group of each atomic site is determined and the respective symmetry operations and LM values of the lattice harmonics representation are printed. The latter information is written into **case.in2_sy**, while the local rotation matrix, the positive or negative IATNR values and the proper ISPLIT parameter are written to **case.struct_st**. (See appendix A and Sec. 4.3).

6.3.1 Execution

The program symmetry is executed by invoking the command:

symmetry symmetry.def or **x symmetry**

6.4 LSTART (atomic LSDA program)

lstart is a relativistic atomic LSDA code originally written by Desclaux (69, 75) and modified for the present purpose. Internally it uses Hartree atomic units, but all output has been converted to Rydberg units. **lstart** generates atomic densities which are used by **dstart** to generate a starting density for a scf calculation and all the input files for the scf run: **in0**, **in1**, **in2**, **inc** and **inm** (according to the atomic eigenvalues). In addition it creates atomic potentials (which are truncated at their corresponding atomic radii and could be used to run **lapw1**) and optional atomic valence densities, which can be used in **lapw5** for a difference density plot. The atomic total energies are also printed, but it can only be used for cohesive energy calculations of light elements. Already for second-row elements the different treatment of relativistic effects in **lstart** and **lapwso** yields inconsistent data and you must calculate the atomic total energy consistently by a supercell approach via a "bandstructure calculation (Put a single atom in a sufficiently large fcc-type unit cell).

If the program stops with some lines:

```
NSTOP= .....
```

in **case.outputst**, this means, that a proper solution for at least one orbital could not be obtained. In such a case the input must be changed and one should provide different occupation numbers for these states (e.g. Cu can not be started with $3d^{10}4s^1$, but it works with $3d^9 4s^2$).

The program produces "WARNINGS" if R0 is too big or core-density leaks out of RMT.

6.4.1 Execution

The program **lstart** is executed by invoking the command:

lstart lstart.def or **x lstart [-sigma]**

The files **case.rsp(up|dn)** are generated and contain the atomic (spin) densities, which will be used by DSTART later on.
Using **-sigma** generates **case.inst_sigma** with modified input to generate **case.sigma** used for difference densities (see below).

6.4.2 Dimensioning parameters

The following parameters are defined in file **param.inc** (static and not allocatable arrays):

NPT	total number of radial mesh points, must be gt.(NRAD+NPT00), where NRAD is the number of mesh-points up to RMT specfied in case.struct.
NPT00	max. number of radial mesh points beyond RMT
RMAX0	max. distance of radial mesh

6.4.3 Input

When running **lstart** you will first be asked interactively to specify an XC-potential switch. Currently 5 (LSDA, Perdew and Wang 92) as well as 11, 13 and 19 (three GGAs, Wu,Cohen 06; the standard "PBE" Perdew et al. 96, as well as "PBEsol", Perdew et al. 08; respectively) are officially supported, 13 is the "standard PBE-GGA".

In addition the program asks for an energy cut-off, separating core from valence states. Usually -6.0 Ry is a good choice, but you should check for each atom how much core charge leaks out of the sphere (WARNINGS in **case.outputs**). If this is the case one should lower this energy cut-off and thus include these low lying states into the valence region. Alternatively you can also select a "charge localization" criterium (usually between 0.97 and 0.9999). This allows a more localized state (like a 4f of 5d elements) to be core, while a more delocalized state at lower energy (like the 5p states of 5d elements) to be semi-core.

The rest of the input is described in the sample input below.

Note: Only the data at the beginning of the line are read whereas the comment describes the respective orbitals. This file can be generated automatically in **w2web** during "*Initialize calc.* or using "*SinglePrograms* ⇩ *instgen_lapw*" or with the script **instgen_lapw**. To edit this file by hand choose "*View/Edit* ⇩ *Input Files*" and choose **case.inst**.

```
------------------ top of file: case.inst --------------------
ZINC
Ne 6          (inert gas, # OF VALENCE ORBITALS not counting spin)
3,-1,1.0  N         ( N,KAPPA,OCCUP; = 3S UP, 1 ELECTRON)
3,-1,1.0  N                            3S DN
3,-2,2.0  N                            3P UP
3,-2,2.0  N                            3P DN
3, 1,1.0  N                            3P*UP
3, 1,1.0  N                            3P*DN
3,-3,3.0  P                            3D UP
3,-3,3.0  P                            3D DN
3, 2,2.0  P                            3D*UP
3, 2,2.0  P                            3D*DN
4,-1,1.0  P                            4S UP
4,-1,1.0  P                            4S DN
****                END OF Input
****                END OF Input
------------------- bottom of file ----------------------------
```

Interpretive comments follow:

line 1: format(a4,a6)
title, keyword

title	
keyword	The keyword **Watson** enables a stabilization of negative ions using a "Watson"-sphere of radius R-wat with charge Q-wat, which must be given in the next line when this keyword is specified. The keyword **PRATT** enables a scf mixing using standard PRATT scheme. It might be usefull if a certain atomic configuration does not converge with the standard mixing scheme and requires a (usually quite small) mixing factor, which must be given in the next line when this keyword is specified.

line 2: free format
config

config — specifies the core state configuration by an inert gas (He, Ne, Ar, Kr, Xe, Rn) and the number of (valence) orbitals (without spin). (In the example given above one could also use **Ar 3** and omit the $3s$ and $3p$ states.) The atomic configurations are listed in the appendix and can also be found online using **`periodic_table`**, a shell script which displays **`SRC/periodic.ps`** with ghostview)

line 3: format(i1,1x,i2,1x,f5.3,a1)
n, kappa, occup, plot

n	the principle quantum number
kappa	the relativistic quantum number (see below)
occup	occupation number (per spin)
plot	**P** specifies that the density of the respective orbital is written to the file **`case.sigma`**, which can be used for difference density plots in **`lapw5`**. **N** or an empty field will exempt density of the respective orbital from being printed to file.

>>>**:line 3** is repeated for the other spin and for all orbitals specified above by config.
>>>**: the last two lines** must be
****
****
optional inserted as line 2 when "Watson" has been specified in line 1: free format
R-wat, Q-wat

R-wat	radius of a charged sphere used to stabilize otherwise unstable negative ions (e.g. 2.5 for O^{2-})
Q-wat	charge of the stabilizing sphere, (e.g. 2 for O^{2-})

The quantum numbers are defined as follows (see e.g. Liberman et al 65):

Spin quantum number: $s = +1$ or $s = -1$

Orbital quantum number $j = l + s/2$

Relativistic quantum number $\kappa = -s(j + 1/2)$

		$j = l + s/2$		κ		max. occupation	
	l	$s = -1$	$s = +1$	$s = -1$	$s = +1$	$s = -1$	$s = +1$
s	0		1/2		-1		2
p	1	1/2	3/2	1	-2	2	4
d	2	3/2	5/2	2	-3	4	6
f	3	5/2	7/2	3	-4	6	8

Table 6.6: Relativistic quantum numbers

6.5 KGEN (generates k mesh)

This program generates the k-mesh in the irreducible wedge of the Brillouin zone (IBZ) on a special point grid, which can be used in a modified tetrahedron integration scheme (Blöchl et al 1994).

`kgen` needs as interactive input the total number of k-points in the BZ. If this is set to zero, you are asked to specify the divisions of the reciprocal unit-cell vectors (3 numbers, be careful not to "break" symmetry and choose them properly according to the inverse lenght of the reciprocal

lattice vectors) for a mesh yourself. If inversion symmetry is not present, it will be added automatically unless you specified the "-so" switch (for magnetic cases with spin-orbit coupling). The k-mesh is then created with this additional symmetry. If symmetry permits, it further asks whether or not the k-mesh should be shifted away from high symmetry directions. The file **case.klist** is used in **lapw1** and **case.kgen** is used in **tetra** and **lapw2**, if the EF switch is set to TETRA, i.e. the tetrahedron method for the k-space integration is used. For the format of the **case.klist** see page 94.

6.5.1 Execution

The program kgen is executed by invoking the command:

kgen kgen.def or **x kgen [-so]**

With the switch **-so** it uses a file **case.ksym** (usually generated by **symmetso**) instead of **case.struct** and does not add inversion symmetry.

6.5.2 Dimensioning parameters

The following parameters are used in **main.f, ord1.f** (static arrays):

IDKP	number of inequivalent k-points (like NKPT in other programs)
NWX	internal parameter, must be increased for very large k-meshes
INDEXM	internal parameter, must be increased for very large k-meshes

6.6 DSTART (superposition of atomic densities)

This program generates an initial crystalline charge density **case.clmsum** by a superposition of atomic densities (**case.rsp**) generated with **lstart**. Information about LM values of the lattice harmonics representation and number of Fourier coefficients of the interstitial charge density are taken from **case.in1** and **case.in2**. In the case of a spin-polarized calculation it must also be run for the spin-up charge density **case.clmup** and spin-down charge density **case.clmdn**.

6.6.1 Execution

The program **dstart** is executed by invoking the command:

dstart dstart.def or **x dstart [-up|dn -c -fft -super -lcore]**

With the switch **-fft** dstart will terminate after **case.in0_std** has been created. The switch **-super** will produce **new_super.clmsum** instead of **case.clmsum**, which is necessary for charge extrapolation (**clmextrapol_lapw**). **-lcore** produces **case.clmsc** from the radial core densities **case.rsplcore**.

6.6.2 Dimensioning parameters

The following parameters are collected in file **module.f**, but usually need not to be changed:

NPT	number of r-mesh points in atomic density (should be the same as in LSTART)
LMAX2	max l in LM expansion
NCOM	number of LM terms in density

7 Programs for running an SCF cycle

Contents

In sections 7.1-7.9 we describe the main programs to run an SCF cycle as illustrated in figure 4.1.

7.1 LAPW0 (generates potential)

`lapw0` computes the total potential V_{tot} as the sum of the Coulomb V_c and the exchange-correlation potential V_{xc} using the total electron (spin) density as input. It generates the spherical part (l=0) as **`case.vsp`** and the non-spherical part as **`case.vns`**. For spin-polarized systems, the spin-densities **`case.clmup`** and **`case.clmdn`** lead to two pairs of potential files. These files are called: **`case.vspup`**, **`case.vnsup`** and **`case.vspdn`**, **`case.vnsdn`**.

The Coulomb potential is calculated by the multipolar Fourier expansion introduced by Weinert (81). Utilizing the spatial partitioning of the unit cell and the dual representation of the charge density [equ. 2.10], firstly the multipole moments inside the spheres are calculated (Q-sp). The Fourier series of the charge density in the interstitial also represent SOME density inside the spheres, but certainly NOT the correct density there. Nevertheless, the multipole moments of this artificial plane-wave density inside each sphere are also calculated (Q-pw). By subtracting Q-pw from Q-sp one obtains pseudo-multipole moments Q. Next a new plane-wave series is generated which has two properties, namely zero density in the interstitial region and a charge distribution inside the spheres that reproduces the pseudo-multipole moments Q. This series is added to the original interstitial Fourier series for the density to form a new series which has two desirable properties: it simultaneously represents the interstitial charge density AND it has the same multipole moments inside the spheres as the actual density. Using this Fourier series the interstitial Coulomb potential follows immediately by dividing the Fourier coefficients by K^2 (up to a constant).

Inside the spheres the Coulomb potential is obtained by a straightforward classical Green's function method for the solution of the boundary value problem.

The exchange-correlation potential is computed numerically on a grid. Inside the atomic spheres a least squares procedure is used to reproduce the potential using a lattice harmonics representation

(the linear equations are solved with modified LINPACK routines). In the interstitial region a 3-dimensional fast Fourier transformation (FFT) is used.

The total potential V is obtained by summation of the Coulomb V_C and exchange-correlation potentials V_{xc}.

In order to find the contribution from the plane wave representation to the Hamilton matrix elements we reanalyze the Fourier series in such a way that the new series represents a potential which is zero inside the spheres but keeps the original value in the interstitial region and this series is put into **case.vns**.

The contribution to the total energy which involves integrals of the form $\rho * V$ is calculated according to the formalism of Weinert et al (82).

The Hellmann-Feynman force contribution to the total force is also calculated (Yu et al 91).

Finally, the electric field gradient (EFG) is calculated in case you have an L=2 term in the density expansion. The EFG tensor is given in both, the "local-rotation-matrix" coordinate system, and then diagonalized. The resulting eigenvectors of this rotation are given by columns.

For surface calculations the total and electrostatic potential at z=0 and z=0.5 is calculated and can be used as energy-zero for the determination of the workfunction. (It is assumed that the middle of your vacuum region is either at z=0 or z=0.5).

7.1.1 Execution

The program **lapw0** is executed by invoking the command:

lapw0 lapw0.def or **x lapw0 [-p -eece -grr]**

7.1.2 Dimensioning parameters

The following parameters are used (they are collected in file **param.inc**, but usually need not to be changed:

NCOM	number of lm components in charge density and potential representation; it must satisfy the following condition: NCOM+3 .gt. {[number of l, m with $m = 0$] + [2 * number of l, m with $m > 0$]}
NRAD	number of radial mesh points
LMAX2	highest L in the LM expansion of charge and potential
LMAX2X	highest L for the gpoint-grid in the xcpot generation (may need large values for "-eece")
restrict_output	for mpi-jobs, limits the number of case.output0xxx files to "restrict_output"

7.1.3 Input

The input is very simple. It is generated automatically by **init_lapw**, and needs to be changed only if a different exchange-correlation potential should be used:

```
------------------ top of file: case.in0 --------------------
TOT   13             ! MULT/COUL/EXCH/POT /TOT ;  VXC-SWITCH
NR2V      IFFT  8    ! R2V  EECE/HYBR IFFT  LUSE
 30  30 108  4.00  1 ! min IFFT-parameters, enhancement factor, iprint
0 0.0     (#of FK in E-field expansion, EFELD (Ry)

------------------- bottom of file ---------------------------
```

Interpretive comments follow:

line 1: free format
switch, indxc

switch		
	TOT	total energy contributions and total potential calculated
	KXC	total energy contributions and total potential calculated. In addition the kinetic energy contribution as well as the XC-energy will be printed.
	POT	total potential is calculated, but not the total energy
	MULT	multipole moments calculated only
	COUL	Coulomb potential calculated only
	EXCH	exchange correlation potential calculated only
		NOTE: MULT, COUL, and EXCH are for testing only, whereas POT, saves some CPU time if total energy is not needed
indxc		index to specify type of exchange and correlation potential. Supported options (for more options see the SRC_lapw0/vxclm2.f subroutine) include:
	5	Perdew and Wang 92, reparameterization of Ceperly-Alder data, the recommended LDA option
	13	Generalized Gradient approximation (PBE, Perdew-Burke-Ernzerhof 96)
	11	Generalized Gradient approximation (Wu-Cohen 2006, Tran et al. 2007)
	19	Generalized Gradient approximation (PBEsol, Perdew 2008)
	20	Generalized Gradient approximation (AM05, Mattsson 2008)
	12	Meta-GGA PKZB (Perdew et al. 1999). In order to generate the requiered **case.vresp*** files, you need **case.inm_vresp** (**cp $WIENROOT/SRC_templates/case.inm_vresp case.inm_vresp** and run one scf cycle with PBE (indxc=13) after creation of **case.inm_vresp**. Only afterwards change indxc to 12. In addition you must use very large IFFT parameters, otherwise it might be numerically unstable.
	27	Meta-GGA TPSS (Tao et al. 2003). At present the "best" meta-GGA. (See also the option above.) Note: Only E_{XC} is obtained that way, V_{XC} of standard PBE is used.
	28	modified Becke-Johnson (mBJ-LDA) potential V_{XC} (Tran and Blaha 2009). Uses a mBJ exchange potential + LDA correlation potential and yields gaps in very good agreement with experiment. The xc-energy E_{XC} is from LDA. For detailed usage see chapter about mBJ calculations (4.5.8).
	50	calculates the average of $\nabla\rho(r)/\rho(r)$ over the unit cell. This is used for the mBJ potential mentioned above.

line 2: free format (only blanks are allowed as separator)
RPRINT, H-mod, FFTopt, LUSE

RPRINT	NR2V	no additional output
	R2V	Exchange-correlation (**case.r2v**), Coulomb (**case.vcoul**) and total potentials (**case.vtotal**) are written as (r^2V) to a file for plotting with **lapw5** (cp **case.vtotal case.clmval**; use "VAL" for normalization in **case.in5**)
H-mod	EECE	On-site Hartree-Fock (inside spheres) for selected electrons (see 4.5.7)
	HYBR	On-site Hybrid functionals (inside spheres) (see 4.5.7)

FFTopt IFFT optional keyword, which lets you define the IFFTx mesh and an enhancement factor in the next line (necessary for **runeece_lapw**)

LUSE optional l-max value for the angular grid used in xcpot1. For standard LDA/GGA the recommended value is max L value of LM-list in case.in2 + 2; for EECE one should use a better, antialiased grid, thus a large negative LUSE-value is recommended (and set automatically by **runeece_lapw**)

line 3: free format (must be omitted when IFFT is not specified above)
IFFTx, IFFTy, IFFTz, IFFTfactor, iprint

IFFTx,y,z FFT-mesh parameters in x,y,z directions for the calculation of the XC-potential in the interstitial region. Usually set automatically in **init_lapw** (dstart). The ratio of the 3 numbers should be indirect proportional to the lattice parameters. (-1 -1 -1 determines these numbers automatically and takes only IFFTfactor into account)

IFFTfactor Multiplicative factor to the IFFT grid specified above. It needs to be enlarged for highly accurate GGA or meta-GGA calculations as well as for systems with H atoms with small spheres.

iprint optional print switch. iprint=0 will greatly reduce case.output0 (in particular for **lapw0_mpi**).

The following line is optional and can be omitted. It is used to introduce an electric field via a zig-zag potential (see J.Stahn et al. 2001):

line 4: free format
IFIELD, EFIELD, WFIELD

IFIELD number of Fourier coefficients to model the zig-zag potential. Typically use IEFIELD=30; -999 lists available modes (form) of fields, and these modes can be specified by mode=IEFIELD/1000. (default: mode=0)

EFIELD value (amplitude) of the electric field. The electric field (in Ry/bohr) corresponds to EFIELD/c, where c is your c lattice parameter.

WFIELD optional value for lambda (see output of IEFIELD=-999).

7.2 ORB (Calculate orbital dependent potentials)

This program was contributed by:

P.Novák
Inst. of Physics, Acad.Science, Prague, Czeck Republic
email: novakp@fzu.cz

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

orb calculates the orbital dependent potentials, i.e. potentials which are nonzero in the atomic spheres only and depend on the orbital state numbers l, m. In the present version the potential is assumed to be independent of the radius vector and needs the density matrix calculated in **lapwdm**. Four different potentials are implemented in this package:

- LDA+U. There are three variants of this method, two of them are discussed in Novák et al. 2001
  1. LDA+U(SIC) - introduced by Anisimov et al. 1993, with an approximate correction for the self-interaction correction. This is probably best suited for strongly correlated systems and for a full potential method we recommend to use an "effective" $U_{eff} = U - J$; setting $J = 0$.
  2. LDA+U(AMF) - introduced by Czyzyk and Sawatzky 1994 as 'Around the Mean Field' method. (In Novák et al. 2001 it is denoted as LDA+U(DFT)). This version is (probably) more suitable for metallic or less strongly correlated systems.
  3. LDA+U(HMF) - in addition the Hubbard model in the mean field approximation, as introduced by Anisimov et al. 1991 is also implemented. Note, however, that it is to be used with the LDA (not LSDA) exchange-correlation potential in spin polarized calculations!

  All variants are implemented in the rotationally invariant way (Liechtenstein et al. 1995). If LDA+U is used in an unrestricted, general way, it introduces an orbital field in the calculation (in analogy to the exchange field in spin-polarized calculations, but it interacts with the orbital, instead of spin momentum). The presence of such an orbital field may lower the symmetry. In particular the complex version of LAPW1 must be used. Care is needed when dealing with the LDA+U orbital field. It may be quite large, and without specifying its direction it may fluctuate, leading to oscillations of scf procedure or/and to false solutions. It is therefore necessary to use it in combination with the spin-orbit coupling, preferably running first LSDA+(s-o) and then slowly switching on the LDA+U orbital field. If the LDA+U orbital polarization is not needed, it is sufficient to run real version of LAPW1, which then automatically puts the orbital field equal to zero. For systems without the center of inversion, when LAPW1 must be complex, an extra averaging of the LDA+U potential is necessary.
- Orbital polarization. The additional potential has the form (Brooks 1985, Eriksson et al. 1989):
  $$V_{OP} = c_{OP} < L_z > l_z \tag{7.1}$$
  where c_{OP} is the orbital polarization parameter, $< L_z >$ is projection of the orbital momentum on the magnetization direction and l_z is single electron orbital momentum component z parallel to $\vec{M}$.
- Exact exchange and Hybrid methods: see Tran et al. 2006 and 4.5.7
- Interaction with the external magnetic field. In this case the additional potential has a simple form:
  $$V_{Bext} = \mu_B \vec{B}_{ext}(\vec{l} + 2\vec{s}). \tag{7.2}$$
  The interaction with the electronic spin is taken into account by shifting the spin up and spin down exchange correlation potentials in LAPW0 by the energy $+\mu_B B_{ext} - \mu_B B_{ext}$, respectively. The interaction of B_{ext} with spin could be as well calculated using the 'Fixed spin moment' method. For an interaction with the orbital momentum it is necessary to specify the atoms and angular momentum numbers for which this interaction will be considered. Caution is needed when considering interaction of the orbital momentum with B_{ext} in metallic or metallic-like systems. For the analysis see the paper by Hirst 1997

In all cases the resulting potential for a given atom and orbital number l is a Hermitian, $(2l + 1)$x$(2l + 1)$ matrix. In general this matrix is complex, but in special cases it may be real.

For more information see also section 4.5.6.

7.2.1 Execution

The program **orb** is executed by invoking the command:

x orb [-up/-dn/-du] or **orb up/dnorb.def**

7.2.2 Dimensioning parameters

The following parameters are used (collected in file **`param.inc`**):

LABC	highest l+1 value of orbital dependent potentials
NRAD	number of radial mesh points

7.2.3 Input

Since this program can handle three different cases, examples and descriptions of **`case.inorb`** for all cases are given below:

Input for all potentials

line 1: free format
nmod,natorb,ipr

nmod	defines the type of potential 1...LDA+U, 2...OP, 3...B_{ext}
natorb	number of atoms for which orbital potential V_{orb} is calculated
ipr	printing option, the larger ipr, the longer the output

line 2: (A5,f8.2)
mixmod,amix

mixmod	PRATT or BROYD (should not be changed, see MIXER for more information)
amix	coefficient for the Pratt mixing of V_{orb} This option is now only used for testing. The mixing should be set to PRATT, 1.0

line 3: free format
iatom(i),nlorb(i),(lorb(li,i),li=1,nlorb(i))

iatom	index of atom in struct file
nlorb	number of orbital moments for which Vorb shall be applied
lorb	orbital numbers (repeated nlorb-times)

3rd line repeated natorb-times

Input for LDA+U (nmod=1)

line 4: free format

nsic	defines 'double counting correction'
nsic=0	'AMF method' (Czyzyk et al. 1994)
nsic=1	'SIC method' (Anisimov et al. 1993, Liechtenstein et al. 1995)
nsic=2	'HMF method' (Anisimov et al. 1991)

line 5: free format

U(li,i), J(li,i)	Coulomb and exchange parameters, U and J, for LDA+U in Ry for atom type i and orbital number li. We recommend to use U_{eff} only.

5th line repeated natorb-times, for each natorb repeated nlorb-times

Example of the input file for NiO (LDA+U included for two inequivalent Ni atoms that have indexes 1 and 2 in the structure file):

```
---------------- top of file: case.inorb --------------------
1  2  0                       nmod, natorb, ipr
PRATT,1.0                     mixmod, amix
1 1 2                         iatom nlorb, lorb
2 1 2                         iatom nlorb, lorb
1                             nsic (LDA+U(SIC) used)
0.52 0.0                      U J
0.52 0.0                      U J
---------------- bottom of file:  --------------------
```

Input for Orbital Polarization (nmod=2)

line 4: (free format)

nmodop		defines mode of 'OP'
	1	average L_z taken separately for spin up, spin down
	0	average L_z is the sum for spin up and spin down

line 5: (free format)

Ncalc(i)		
	1	Orb.pol. parameters are calculated ab-initio
	0	Orb.pol. parameters are read from input

this line is repeated natorb-times
line 6: (free format) (only if Ncalc=0, then repeated nlorb-times)

pop(li,i)	OP parameter in Ry

line 7: (free format)
xms(1), xms(2), xms(3)

direction of magnetization expressed in terms of lattice vectors

Example of the input file for NiO (total $< L_z >$ used in (1), OP parameters calculated ab-initio, $\vec{M}$ along [001]):

```
---------------- top of file: case.inorb --------------------
2  2  0                       nmod, natorb, ipr
PRATT, 1.0                    mixmod, amix
1 1 2                         iatom nlorb, lorb
2 1 2                         iatom nlorb, lorb
0                             nmodop
1                             Ncalc
1                             Ncalc
0. 0. 1.                      direction of M in terms of lattice vectors
---------------- bottom of file --------------------
```

Input for interaction with B_{ext} (nmod=3)

line 4: (free format)

B_{ext}	external field in Tesla

line 5: (free format)
xms(1), xms(2), xms(3)

direction of magnetization expressed in terms of lattice vectors

Example of the input file for NiO, (B_{ext}= 4 T, along [001]):

```
---------------- top of file: case.inorb --------------------
3  2  0                       nmod, natorb, ipr
PRATT, 1.0                    mixmod, amix
1 1 2                         iatom nlorb, lorb
2 1 2                         iatom nlorb, lorb
4.                            Bext in T
0. 0. 1.                      direction of Bext in terms of lattice vectors
---------------- bottom of file --------------------
```

7.3 LAPW1 (generates eigenvalues and eigenvectors)

lapw1 sets up the Hamiltonian and the overlap matrix (Koelling and Arbman 75) and finds by diagonalization eigenvalues and eigenvectors which are written to **case.vector**. Besides the standard LAPW basis set, also the APW+lo method (see Sjöstedt et al 2000, Madsen et al. 2001) is supported and the basis sets can be mixed for maximal efficiency. If the file **case.vns** exists (i.e. non-spherical terms in the potential), a full-potential calculation is performed.

For structures without inversion symmetry, where the hamilton and overlap matrix elements are complex numbers, the corresponding program version **lapw1c** must be used in connection with **lapw2c**.

Since usually the diagonalization is the most time consuming part of the calculations, two options exist here. In **WIEN2k** we include highly optimized modifications of LAPACK routines. We call all these routines "full diagonalization", but we also provide an option to do an "iterative diagonalization" using a new preconditioning of a block-Davidson method (see Singh 89 and Blaha et al. 09). The scheme uses an old eigenvector from the previous scf-iteration, and produces approximate (but usually still highly accurate) eigenvalues/vectors. The preconditioner (inverse of $(H - \lambda S)$ can be calculated at the first iterative step (which will therefore take longer than subsequent iterative steps), stored on disk (**case.storeHinv**) and reused in all subsequent scf-iterations (until the next "full" diagonalization or when it is recreated (x lapw1 -it -noHinv0)). Usually this is the fastest scheme, but storage of **case.storeHinv** can be large (and slow when you have a slow network) and when the Hamiltonian changes too much, performance may degrade. Alternatively, the preconditioner can be recalculated all the time (x lapw1 -it -noHinv). Depending on the ratio of matrix size to number of eigenvalues (cpu time increases linearly with the number of eigenvalues, but a sufficiently large number is necessary to ensure convergence) a significant speedup compared to "full" diagonalization (LAPACK) can be reached. Iterative diagonalization is activated with the **-it** switch in **x lapw1 -it** or **run_lapw -it**. Often the preconditioner is so efficient, that it does not need to be recalculated, even within a structural optimization and one can use **min_lapw -j ``run_lapw -I -fc 1 -it''**. In some cases it is preferable to use **min_lapw -j ``run_lapw -I -fc 1 -it1''**, which will recreate **case.storeHinv**, or do not store these files at all using **min_lapw -j ``run_lapw -I -fc 1 -it -noHinv ''**

Parallel execution (fine grain and on the k-point level) is also possible and is described in detail in Sec. 5.5.

The switch **-nohns** skips the calculation of the nonspherical matrix elements inside the sphere. This may be used to save computer time during the first scf cycles.

7.3.1 Execution

The program **lapw1** is executed by invoking the command:

x lapw1 [-c -up|dn -it -noHinv|-noHinv0 -p -nohns -orb -band -nmat_only] or

lapw1 lapw1.def or **lapw1c lapw1.def**

In cases without inversion symmetry, the default input filename is **case.in1c**. For 2-window (not recommended) semi-core calculations the **lapw1s.def** file uses a **case.in1s** file and creates the files **case.output1s** and **case.vectors**. For the spin-polarized case lapw1 is called twice with **uplapw1.def** and **dnlapw1.def**. To all relevant files the keywords "**up**" or "**dn**" are appended (see the fcc Ni test case in the **WIEN2k** package).

7.3.2 Dimensioning parameters

The following parameters (collected in file **param.inc_r** or **param.inc_c**) are used:

LMAX	highest l+1 in basis function inside sphere (consistent with input in case.in1)
LMMX	number of LM terms in potential (should be at least NCOM-1)
LOMAX	highest l for local orbital basis (consistent with input in case.in1)
NGAU	number of Gaunt coefficients for the non-spherical contributions to the matrix elements
NMATMAX	maximum size of H,S-matrix (defines size of program, should be chosen according to the memory of your hardware, see chapter 11.2.2!)
NRAD	number of radial mesh points
NSLMAX	highest l+1 in basis functions for non-muffin-tin matrix elements (consistent with input in case.in1).If set larger than 5, parameter MAXDIM (modules.F) and LOMAX=8, P(10,10) (gaunt2.f) must also be increased.
NSYM	order of point group
NUME	maximum number of energy eigenvalues per k-point
NVEC1	defines the largest triple of integers which define reciprocal
NVEC2	K-vectors when multiplied with the reciprocal Bravais matrix
NVEC3	
NLOAT	max number of LOs for one l-quantum number
restrict_output	for mpi-jobs, limits the number of case.output1_X_proc_XXX files to "restrict_output"

7.3.3 Input

Below a sample input is shown for TiO_2 (rutile), one of the test cases provided in the **WIEN2k** package. The input file is written automatically by LSTART, but was modified to set APW only for Ti-3d and O-2p orbitals.

```
------------------ top of file: case.in1 --------------------
WFFIL  EF=0.5000   (WFPRI,WFFIL,SUPWF ; wave fct. print,file,suppress
  7.500   10    4  (R-mt*K-max; MAX l, max l for hns )
  0.30    5 0 (global energy parameter E(l), with 5 other choices,   LAPW)
 0   -3.00      0.020 CONT 0   ENERGY PARAMETER for s,               LAPW
 0    0.30      0.000 CONT 0   ENERGY PARAMETER for s-local orbital, LAPW-LO
 1   -1.90      0.020 CONT 0   ENERGY PARAMETER for p               LAPW
 1    0.30      0.000 CONT 0   ENERGY PARAMETER for p-local orbitals LAPW-LO
 2    0.20      0.020 CONT 1                                        APW
  0.20    3 0 (global energy parameter E(l), with 1 other choice,    LAPW)
 0   -0.90      0.020 STOP 0                                        LAPW
```

```
 0    0.30      0.000 CONT 0                                          LAPW-LO
 1    0.30      0.000 CONT 1                                          APW
K-VECTORS FROM UNIT:4   -9.0       2.0    69   emin/emax/nband
1.d-15    0.0                   spro_limit for it.diag., lambda for it.diag
------------------- bottom of file ------------------------
```

Interpretive comments follow:

line 1: free format
switch, EF

switch	WFFIL	standard option, writes wave functions to file **`case.vector`** (needed in **`lapw2`**)
	SUPWF	suppresses wave function calculation (faster for testing eigenvalues only)
	WFPRI	prints eigenvectors to **`case.output1`** and writes **`case.vector`** (produces long outputs!)
EF		optional input. If "EF=" key is present, lapw1 reads EF and sets all default energy parameters (0.3) to "EF-0.2" Ry.

line 2: free format
rkmax, lmax, lnsmax

rkmax	$R_{mt} * K_{max}$ determines matrix size (convergence), where Kmax is the plane wave cut-off, Rmt is the smallest of all atomic sphere radii. Usually this value should be between 5 and 9 (APW+lo) or 6 - 10. (LAPW-basis) (K^2_{max} would be the plane wave cut-off parameter in Ry used in pseudopotential calculations.) Note that d (f) wavefunctions converge slower than s and p. For systems including hydrogen with short bondlength and thus a very small R_{mt} (e.g. 0.7 a.u.), RKmax = 3 might already be reasonable, but convergence must be checked for a new type of system. Note, that the actual matrix size is written on case.scf1. It is determined by whatever is smaller, the plane wave cut-off (specified with RKmax) or the maximum matrix dimension NMATMAX, (see previous section).
lmax	maximum l value for partial waves used inside atomic spheres (should be between 8 and 12)
lnsmax	maximum l value for partial waves used in the computation of non-muffin-tin matrix elements (lnsmax=4 is quite good)

line 3: free format
Etrial, ndiff, Napw

Etrial	default energy used for all E_l to obtain $u_l(r, E_l)$ as regular solution of the radial Schrödinger equation [used in equ.2.4,2.7] (see figure 7.1).
ndiff	number of exceptions (specified in the next ndiff lines)
Napw	0 ... use LAPW basis, 1 ... use APW-basis for all "global" l values of this atom. We recommend to use LAPW here.

line 4: format(I2,2F10.5,A4)
l, El, de, switch, NAPWL

l	l of partial wave
El	E_l for L=l
de	energy increment

de=0: this E(l) overwrites the default energy (from line 3)
de$\neq$ 0: a search for a resonance energy using this increment is done. The radial function $u_l(r, E)$ up to the muffin-tin radius RMT varies with the energy. A typical case is schematically shown in Fig. 7.1.
At the bottom of the energy bands u has a zero slope (bonding state), but it has a zero value (antibonding state) at the top of the bands. One can search up and down in energy starting with E_l using the increment de to find where $u_l(R_{MT}, E)$ changes sign in value to determine E_{top} and in slope to specify E_{bottom}. If both are found E_l is taken as the arithmetic mean and replaces the trial energy. Otherwise E_l keeps the specified value. For E_{top} and E_{bottom} bounds of +1 and -10 Ry are defined respectively, and if they are not found, they remain at the initial value set to -200.

switch — used only if de.ne.0
- CONT — calculation continues, even if either E_{top} or E_{bottom} are not found
- STOP — calculation stops if not both E_{top} and E_{bottom} are found (especially useful for semi-core states)

NAPWL — 0 ... use LAPW basis, 1 ... use APW-basis for this l value of this atom. We recommend to use APW+lo when the corresponding wavefunction is "localized" and thus difficult to converge with standard LAPW (like 3d functions) and/or when the respective atomic sphere is small compared to the other spheres in the unit cell.

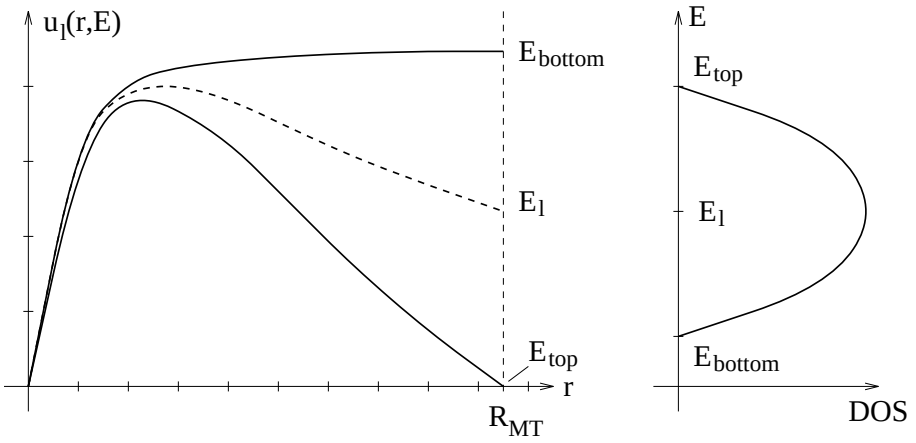

Figure 7.1: Schematic dependence of DOS and $u_l(r, E_l)$ on the energy

>>>**:line 4** is repeated ndiff times (see line 3) for each exception. If the same l value is specified twice, local orbitals are added to the (L)APW basis. The first energy (E_1) is used for the usual LAPW's and the second energy (E_2) for the LOs, which are formed according to (see equ. 2.7): $u_{E_1} + \dot{u}_{E_1} + u_{E_2}$.
*Note: The default energy parameters (**0.30**) are replaced by an energy "$E_F - 0.2$" if the EF-switch was read before. Please read also the comments about **run_lapw** in section 5.1.3. In addition, you may want to change the automatically created input and add d- or f-local orbitals to reduce the linearization error (e.g. in late transition metals you could put E_{3d} at 0.0 and 1.0 Ry) or s, p, d, and/or f-LOs at very high energy (e.g. 2.0 - 3.0 Ry) to better describe unoccupied states.*

>>>**:lines 3 and 4** are repeated for each non equivalent atom

line 5: format (20x,i1,2f10.1,i6)
unit-number, Emin, Emax nband

unit-number	file number from which the k-vectors in the irreducible wedge of the Brillouin zone are read. Default is 4, for which the corresponding information is contained in **case.klist** (generated by KGEN). Should not be changed.
EMIN, EMAX	energy window in which eigenvalues shall be searched (overrides setting in **case.klist**. A small window saves computer time, but it also limits the energy range for the DOS calculation of unoccupied states.
nband	number of eigenvalues calculated with iterative diagonalization. Set automatically to $nband = ne * 2.0 + 5$ in **lstart**. Larger values will lead to more cpu-time. (Optional input)

line 6: free format; optional input line, but necessary if k-vectors are read from unit 5
spro_limit, lambda_iter

spro_limit	limit for detection of linear dependency for iterative diagonalization (optional input), typical around 1.d-15)
lambda_iter	optional λ value for preconditioner of iterative diagonalization (see above). By default we use $\lambda = 0$, but in some cases convergence can be improved by a small (around 1.0) positive or negative λ

line 7: format (A10,4I10,3F5.2); (only when unit-number=5, not recommended, use unit 4 and **case.klist**)
name, ix,iy,iz, idv, weight

name	name of k-vector (optional) >>>: the last line must be END !!
ix,iy,iz, idv	defines the k-vector, where x= ix/idv etc. We use carthesian coordinates in units of $2\pi/a$, $2\pi/b$, $2\pi/c$ for P,C,F and B cubic, tetragonal and orthorhombic lattices, but internal coordinates for H and monoclinic/triclinic lattices
weight	of k-vector (order of group of k)

>>>**: line 7** is repeated for each k-vector in the IBZ. The utility program **kgen** (see section 6.5) provides a list of such vectors (on a tetrahedral mesh) in **case.klist**.
>>>**: the last line** must be END

7.4 LAPWSO (adds spin orbit coupling)

lapwso includes spin-orbit (SO) coupling in a second-variational procedure and computes eigenvalues and eigenvectors (stored in **case.vectorso**) using the scalar-relativistic wavefunctions from **lapw1**. For reference see Singh 94 and Novák 97. The SO coupling must be small, as it is diagonalized in the space of the scalar relativistic eigenstates. For large spin orbit effects it might be necessary to include many more eigenstates from **lapw1** by increasing EMAX in **case.in1** (up to 10 Ry!). We also provide an additional basisfunction, namely an LO with a $p_{1/2}$ radial wavefunction, which improves the basis and removes to a large degree the dependency of the results on EMAX and RMT (see Kuneš et al. 2001). SO is considered only within the atomic spheres and thus the results may depend to some extent on the choice of atomic spheres radii. The nonspherical potential is neglected when calculating $\frac{dV}{dr}$. Orbital dependend potentials (LDA+U, EECE or OP) can be added to the hamiltonian in a cheap and simple way.

In spin-polarized calculations the presence of spin-orbit coupling may reduce symmetry and even split equivalent atoms into non-equivalent ones. It is then necessary to consider a larger part of the Brillouin zone and the input for **lapw2** should also be modified since the potential has lower symmetry than in the non-relativistic case. The following inputs may change:

- **case.struct**
- **case.klist**
- **case.kgen**
- **case.in2c**
- **case.in1**

We recommend to use **initso** (see Sec.5.2.17) which helps you together with **symmetso** (see Sec.9.1) to setup spinorbit calculations.

*Note: SO eigenvectors are complex and thus **lapw2c** must be used in a selfconsistent calculation.*

7.4.1 Execution

The program **lapwso** is executed by invoking the command:

x lapwso [-up -p -c -orb] or
lapwso lapwso.def

where here -up indicates a spin-polarized calculation (no "-dn" is needed, since spin-orbit will mix spin-up and dn states in one calculation).

7.4.2 Dimensioning parameters

The following parameters are used (collected in file **param.inc**):

FLMAX	constant = 3
LMAX	highest l of wave function inside sphere (consistent with **lapw1**)
LABC	highest l of wave function inside sphere where SO is considered
LOMAX	max l for local orbital basis
NRAD	number of radial mesh points
NLOAT	number of local orbitals

7.4.3 Input

A sample input for lapwso is given below. It will be generated automatically by **initso**

```
------------------ top of file: case.inso --------------------
WFFIL
 4  0  0                      llmax,ipr,kpot
 -10.0000   1.5000           Emin, Emax
     0 0 1                      h,k,l (direction of magnetization)
   2                            number of atoms with RLO
 1    -3.5      0.005 STOP      atom-number, E-param for RLO
 3    -4.5      0.005 STOP      atom-number, E-param for RLO
 1   2                          number of atoms without SO, atomnumbers
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

line 1: format(A5)
switch

WFFIL		wavefunctions will also be calculated for scf-calculation. Otherwise only eigenvalues are calculated.

line 2: free format
LLMAX, IPR, KPOT

LLMAX		maximum l for wavefunctions
IPR		print switch, larger numbers give additional output.
KPOT	0	V(dn) potential is used for $< dn\|V\|dn >$ elements, V(up) for $< up\|V\|up >$ and [V(dn)+V(up)]/2 for $< up\|V\|dn >$.
	1	averaged potential used for all matrix elements.

line 3: free format
Emin, Emax

Emin	minimum energy for which the output eigenvectors and eigenenergies will be printed (Ry)
Emax	maximum energy

line 4: free format

h,k,l	vector describing the direction of magnetization. For R lattice use h,k,l in rhombohedral coordinates (not in hexagonal)

line 5: free format

nlr	number of atoms for which a $p_{1/2}$ LO should be added

line 6: free format
nlri, El, de, switch

nlri		atom-number for which RLO should be added
El		E_l for L=l
de		energy increment (see **lapw1**)
switch		used only if de.ne.0
	CONT	calculation continues, even if either E_{top} or E_{bottom} are not found

STOP calculation stops if not both E_{top} and E_{bottom} are found (especially useful for semi-core states)

>>>: line 6 must be repeated "nlr" times (or should be omitted if nlr=0).
line 7: free format
noff, (iatoff(i),i=1,noff)

noff number of atoms for which SO is switched off (for "light" elements, saves time)
iatoff atom-numbers

7.5 LAPW2 (generates valence charge density expansions)

lapw2 uses the files **case.energy** and **case.vector** and computes the Fermi-energy (for a semiconductor E_F is set to the valence band maximum) and the expansions of the electronic charge densities in a representation according to eqn. 2.10 for each occupied state and each k-vector; then the corresponding (partial) charges inside the atomic spheres are obtained by integration. In addition "Pulay-corrections" to the forces at the nuclei are calculated here. For systems without inversion symmetry you have to use the program **lapw2c** (in connection with **lapw1c**).

The partial charges for each state (energy eigenvalue) and each k-vector can be written to files **case.help031**, **case.help032** etc., where the last digit gives the atomic index of inequivalent atoms (switch **-help_files**). Optionally these partial charges are also written to **case.qtl** (switch **-qtl**). For meta-GGA calculations energy densities are written to **case.vrepval**(switch **-vresp**).

In order to get partial charges for bandstructure plots, use **-band**, which sets the "QTL option and uses "ROOT" in **case.in2**. Several other switches change the input file **case.in2** temporarely and are described there.

7.5.1 Execution

The program **lapw2** is executed by invoking the command:

x lapw2 [-c -up|dn -p -so -qtl -fermi -efg -band -eece -vresp -help_files -emin X -all X Y] or
lapw2 lapw2.def [proc#] or **lapw2c lapw2.def [proc#]**

where proc# is the i-th processor number in case of parallel execution (see Fig. 5.2). The **-so** switch sets **-c** automatically.

For complex calculations `case.in2c` is used. For a spin-polarized case see the fcc Ni test case in the **WIEN2k** package.

7.5.2 Dimensioning parameters

The following parameters are used (collected in file modules.F):

IBLOCK Blocking parameter (32-255) in **l2main.F**, optimize for best performance
LMAX2 highest l in wave function inside sphere (smaller than in **lapw1**, at present must be .le. 8)
LOMAX max l for local orbital basis

NCOM	number of LM terms in density
NGAU	max. number of Gaunt numbers
NRAD	number of radial mesh points
restrict_output	for mpi-jobs, limits the number of case.output2_X_proc_XXX files to "restrict_output"

7.5.3 Input

A sample input for **lapw2** is listed below, it is generated automatically by the programs **lstart** and **symmetry**.

```
------------------ top of file: case.in2 --------------------
TOT                          (TOT,FOR,QTL,EFG)
-1.2       32.000   0.5  0.05  (EMIN, # of electrons,ESEPERMIN, ESEPER0 )
TETRA       0.0      (EF-method (ROOT,TEMP,GAUSS,TETRA,ALL),value)
  0 0  2 0  2 2  4 0  4 2  4 4
  0 0  1 0  2 0  2 2  3 0  3 2  4 0  4 2  4 4
14.0      (GMAX)
FILE      (NOFILE, optional)
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

line 1: format(2A5)
switch, EECE

switch	TOT	total valence charge density expansion inside and outside spheres
	FOR	same as TOT, but in addition a "Pulay" force contribution is calculated (this option costs extra computing time and thus should be performed only at the final scf cycles, see **run_lapw** script in sec. 5.1.3)
	QTL	partial charges only (generates file **case.qtl** for DOS calculations), set automatically by switch **-qtl**
	EFG	computes decomposition of electric field gradient (EFG), contributions from inside spheres (the total EFG is computed in **lapw0**), set automatically by switch **-efg**.
	ALM	this generates two files, **case.radwf** and **case.almblm**, where the radial wavefunctions and the A_{lm}, B_{lm}, C_{lm} coefficients of the wavefunction inside spheres are listed. The file case.almblm can get very big.
	CLM	CLM charge density coefficients only
	FERMI	Fermi energy only, this produces weight files for parallel execution and for the **optics** and **lapwdm** package, set automatically by switch **-fermi**.
	>>>:	TOT and FOR are the standard options, QTL is used for density of states (or energy bandstructure) calculations, EFG for analysis, while FOURI, CLM are for testing only.
EECE		if set to "EECE", calculates the density for specified atoms and angular momentum only. Used for exact-exchange or hybrid-calculations, usually set automatically by **runsp_lapw -eece**

line 2: free format
emin, ne, esepermin, eseper0

emin	lower energy cut-off for defining the range of occupied states, can be set termporarely to "X" by switch **-emin X** or **-all X Y**

ne	number of electrons (per unit cell) in that energy range
esepermin	LAPW2 tries to find the "mean" energies for each l channel, for both the valence and the semicore states. To define "valence" and "semicore" it starts at (EF - "esepermin") and searches for a "gap" with a width of at least "eseper0" and defines this as separation energy of valence and semicore
eseper0	minimum gap width (see above). The values esepermin and eseper0 will only influence results if the option -in1new is used

line 3: format(a5,f10.5)
efmod, eval

efmod		determines how E_F is determined
	ROOT	E_F is calculated and k space integration is done by root sampling (this can be used for insulators, but for metals poor convergence is found)
	TEMP	E_F is calculated where each eigenvalue is temperature broadened using a Fermi function with a broadening parameter of eval Ry. The total energy is corrected corresponding to T=0K. (e.g. eval=0.002 Ry gives good total energy convergence, but has no "physical" justification)
	TEMPS	E_F is calculated where each eigenvalue is temperature broadened using a Fermi function with a broadening parameter of eval Ry. The total energy is corrected by -TS corresponding to the temperature specified by eval (e.g. eval=0.002 Ry corresponds to about 40 C)
	GAUSS	E_F is calculated as above but a Gaussian smearing method is used with a width of eval Ry. (e.g. eval=0.002 gives good total energy convergence, but has no "physical" justification).
	TETRA	E_F is calculated and k space integration is done by the modified (if eval is .eq. 0) or standard (eval .ge. 100) tetrahedron-method (Blöchl 94). This "standard" scheme is recommended for **optic**. In this case the file **case.kgen**, consistent with the k-mesh used in **lapw1**, must be provided (see Sec. 7.3). This is the recommended option although convergence may be slower than with Gauss- or temperature-smearing.
	ALL	All states up to eval are used. This can be used to generate charge densities in a specified energy interval, can be set termporarely by switch **-all X Y**.
eval		when efmod is set to TEMP(S) (eval=0 will lead to room temperature broadening, 0.0018 Ry) or GAUSS, eval specifies the width of the broadening (in Ry), if efmod is set to ALL, eval specifies the upper limit of the energy window (in Ry; can be set termporarely by switch **-all X Y**), if efmod is set to TETRA, eval .ge. 100 specifies the use of the standard tetrahedron method instead of the modified one (see above). By default, TETRA will average over partially occupied degenerate states at EF with a degeneracy criterium D = 1.d-6. You can modify this by setting eval equal to your desired D (or 100+D).

optional line 3a: free format (ONLY when EECE is set)

nat_rho	number of atoms for which a specific density should be calculated

optional line 3b: free format (ONLY when EECE is set)
jatom_rho, l_rho

jatom_rho	index of atom for which a specific density should be calculated

l_rho — angular momentum l-value for which a specific density should be calculated

>>>**line 3b:** must be repeated nat_rho times
line 4: format (121(I3,I2))

L,M — LM values of lattice harmonics expansion (equ. 2.10), defined according to the point symmetry of the corresponding atom; generated in SYMMETRY, MUST be consistent with the local rotation matrix defined in **`case.struct`** (details can be found in Kara and Kurki-Suonio 81). CAUTION: additional LM terms which do not belong to the lattice harmonics will in general not vanish and thus such terms must be omitted. Automatic termination of the lm series occurs when a second 0,0 pair appears within the list. When you change the l, m list during an SCF calculation the Broyden-Mixing is restarted in MIXER.

>>>**line 4:** must be repeated for each inequivalent atom

Symmetry	LM combinations
23	0 0, 4 0, 4 4, 6 0, 6 4,-3 2, 6 2, 6 6,-7 2,-7 6, 8 0, 8 4, 8 8,-9 2,-9 6,-9 4,-9 8,10 0, 10 4,10 8, 10 2, 10 6, 10 10
M3	0 0, 4 0, 4 4, 6 0, 6 4, 6 2, 6 6, 8 0, 8 4, 8 8,10 0, 10 4,10 8, 10 2, 10 6, 10 10
432	0 0, 4 0, 4 4, 6 0, 6 4, 8 0, 8 4, 8 8,-9 4,-9 8,10 0, 10 4,10 8
-43M	0 0, 4 0, 4 4, 6 0, 6 4,-3 2,-7 2,-7 6, 8 0, 8 4, 8 8,-9 2,-9 6,10 0, 10 4,10 8
M3M	0 0, 4 0, 4 4, 6 0, 6 4, 8 0, 8 4, 8 8,10 0, 10 4,10 8

Table 7.41: LM combinations of "Cubic groups" (3‖(111)) direction, requires "positive atomic index" in case.struct. Terms that should be combined (Kara and Kurki-Suonio 81) must follow one another.

Symmetry	Coordinate axes	Indices of $Y_{\pm LM}$	crystal system
1	any	ALL (±l,m)	triclinic
-1	any	(±2l,m)	
2	2‖ z	(±l,2m)	monoclinic
M	m⊥z	(±l,l-2m)	
2/M	2‖z, m⊥z	(±2l,2m)	
222	2‖z, 2‖y, (2‖x)	(+2l,2m), (-2l+1,2m)	orthorhombic
MM2	2‖z, m⊥y, (2⊥x)	(+l,2m)	
MMM	2⊥z, m⊥y, 2⊥x	(+2l,2m)	
4	4‖z	(±l,4m)	tetragonal
-4	-4‖z	(±2l,4m), (±2l+1,4m+2)	
4/M	4‖z, m⊥z	(±2l,4m)	
422	4‖z, 2‖y, (2‖x)	(+2l,4m), (-2l+1,4m)	
4MM	4‖z, m⊥y, (2⊥x)	(+l,4m)	
-42M	-4‖z, 2‖x (m=xy→yx)	(+2l,4m), (-2l+1,4m+2)	
4MMM	4‖z, m⊥z, m⊥x	(+2l,4m)	
3	3‖z	(±l,3m)	rhombohedral
-3	-3‖z	(±2l,3m)	
32	3‖z, 2‖y	(+2l,3m), (-2l+1,3m)	
3M	3‖z, m⊥y	(+l,3m)	
-3M	-3‖z, m⊥y	(+2l,3m)	
6	6‖z	(±l,6m)	hexagonal
-6	-6‖z	(+2l,6m), (±2l+1,6m+3)	
6/M	6‖z, m⊥z	(±2l,6m)	
622	6‖z, 2‖y, (2‖x)	(+2l,6m), (-2l+1,6m)	
6MM	6‖z, m‖y, (m⊥x)	(+l,6m)	
-62M	-6‖z, m⊥y, (2‖x)	(+2l,6m), (+2l+1,6m+3)	
6MMM	6‖z, m⊥z, m⊥y, (m⊥x)	(+2l,6m)	

Table 7.42: LM combination and local coordinate system of "non-cubic groups" (requires "negative atomic index" in case.struct)

line 5: free format

GMAX max. G (magnitude of largest vector) in charge density Fourier expansion. For systems with short H bonds larger values (e.g. GMAX up to 25) could be necessary. Calculations using GGA (potential option 13 or 14 in case.in0) should also employ a larger GMAX value (e.g. 14), since the gradients are calculated numerically on a mesh determined by GMAX. When you change GMAX during an scf calculation the Broyden-Mixing is restarted in **mixer**.

line 6: A4

reclist FILE writes list of K-vectors into file `case.recprlist` or reuses this list if the file exists. The saved list will be recalculated whenever GMAX, or a lattice parameter has been changed.

NOFILE always calculate new list of K-vectors

7.6 SUMPARA (summation of files from parallel execution)

sumpara is a small program which (in parallel execution of **WIEN2k**) sums up the densities (**case.clmval_***) and quantities from the **case.scf2_*** files of the different parallel runs.

7.6.1 Execution

The program **sumpara** is executed by invoking the 2 commands as follows:

x sumpara -d [-up/-dn/-du] and then
sumpara sumpara.def #_of_proc

where **#_of_proc** is the numbers of parallel processors used. It is usually called automatically from **lapw2para** or **x lapw2 -p**.

7.6.2 Dimensioning parameters

The following parameters are listend in file **param.inc**, but usually they need not to be modified:

NCOM	number of LM terms in density
NRAD	number of radial mesh points
NSYM	order of point group

7.7 LAPWDM (calculate density matrix)

This program was contributed by:

J.Kuneš and P.Novák
Inst. of Physics, Acad.Science, Prague, Czeck Republic
email: novakp@fzu.cz

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

lapwdm calculates the density matrix needed for the orbital dependent potentials generated in **orb**. Optionally it also provides orbital moments, orbital and dipolar contributions to the hyperfine field (only for the specified atoms AND orbitals). It calculates the average value of the operator X which behaves in the same way as the spin-orbit coupling operator: it must be nonzero only within the atomic spheres and can be written as a product of two operators - radial and angular:

$X = X_r(r) * X_{ls}(\vec{l}, \vec{s})$

$X_r(r)$ and $X_{ls}(\vec{l}, \vec{s})$ are determined by RINDEX and LSINDEX in the input as described below:

- RINDEX=0 LSINDEX=0: the density matrix is calculated (this is needed for LDA+U calculations)
- RINDEX=1 LSINDEX=1: <X> is number of electrons inside the atomic sphere (for test)
- RINDEX=2 LSINDEX=1: <X> is the $< 1/r^3 >$ expectation value inside the atomic sphere
- RINDEX=1 LSINDEX=2: <X> is the projection of the electronic spin inside the atomic sphere (must be multiplied by g=2 to get the spin moment)
- RINDEX=1 LSINDEX=3: <X> is the projection of the orbital moment inside the atomic sphere (in case of SO-calculations WITHOUT LDA+U)
- RINDEX=3 LSINDEX=3: <X> is the orbital part of the hyperfine field at the nucleus (for a converged calculation at the very end)
- RINDEX=3 LSINDEX=5: <X> is the spin dipolar part of the hyperfine field at the nucleus (for a converged calculation at the very end)

To use the different operators, set the appropriate input. More information and extentions to operators of similar behavior may be obtained directly from P. Novák (2006). (RINDEX=3 includes now an approximation to the relativistic mass enhancement and LSINDEX=5 includes nondiagonal terms)

lapwdm needs the occupation numbers, which are calculated in **lapw2**. Note: You must not use ROOT in **case.in2** for that purpose.

7.7.1 Execution

The program **lapwdm** is executed by invoking the command:

x lapwdm [-up/dn -p -c -so] or
lapwdm lapwdm.def

7.7.2 Dimensioning parameters

The following parameters are used (collected in file **param.inc**):

FLMAX	constant = 3
LMAX	highest l of wave function inside sphere (consistent with **lapw1**)
LABC	highest l of wave function inside sphere where SO is considered
LOMAX	max l for local orbital basis
NRAD	number of radial mesh points

7.7.3 Input

A sample input for **lapwdm** is given below.

```
------------------ top of file: case.indm --------------------
-9.                     Emin cutoff energy
 1                      number of atoms for which density matrix is calculated
 1  1  2      index of 1st atom, number of L's, L1
 0 0          r-index, (l,s)-index
------------------- bottom of file -----------------------
```

Interpretive comments on this file are as follows:

line 1: free format

emin	lower energy cutoff (usually set to very low number).

line 2: free format

natom	number of atoms for which the density matrix is calculated

line 3: free format
iatom, nl, l

iatom	index of atom for which the density matrix should be calculated
nl	number of l-values for which the density matrix should be calculated
l	l-values for which the density matrix should be calculated

line 3 is repeated natom times t
line 4: free format, optional
RINDEX, LSINDEX

RINDEX 0-3, as described in the introduction to **lapwdm**
LSINDEX0-5, as described in the introduction to **lapwdm**

7.8 LCORE (generates core states)

lcore is a modified version of the Desclaux (69, 75) relativistic LSDA atomic code. It computes the core states (relativistically including SO, or non-relativistically if "NREL" is set in **case.struct**) for the current spherical part of the potential (**case.vsp**). It yields core eigenvalues, the file **case.clmcor** with the corresponding core densities, and the core contribution to the atomic forces.

7.8.1 Execution

The program **lcore** is executed by invoking the command:

lcore lcore.def or **x lcore [-up|-dn]**

For the spin-polarized case see fcc Ni on the distribution tape. If **case.incup** and **case.incdn** are present for spin-polarized calculations, different core-occupation ("open core" approximation for 4f states or spin-polarized core-holes) for both spins are possible.

7.8.2 Dimensioning parameters

The following parameter is listend in file **param.inc**:

NRAD	number of radial mesh points

7.8.3 Input

Below is a sample input (written automatically by **lstart**)

for TiO_2 (rutile), one of the test cases provided with the **WIEN2k**

package.

In case of a "open core" calculation (eg. for 4f states) you may need "spin-polarized" **case.inc** files in order to define different configurations for spin-up and dn. Create two files **case.incup** and **case.incdn** with the corresponding occupations. The **runsp_lapw** script will automatically copy the corresponding files to **case.inc**.

```
------------------ top of file: case.inc --------------------
 4 0.0 0     # of orbitals,  shift of potential, print switch
 1,-1,2      n (principal quantum number), kappa, occup. number
 2,-1,2      2s
 2,-2,4      2p
 2, 1,2      2p*
 1   0.0          # of orbital of second atom
 1,-1,2      1s
 0           end switch
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

line 1: free format
nrorb, shift, iprint

nrorb	number of core orbitals
shift	shift of potential for "positive" eigenvalues (e.g. for 4f states as core states in lanthanides)
iprint	optional print switch to reduce (0) or increase (1) printing to **case.outputc**

line 2: free format
n, kappa, occup

n	principle quantum number
kappa	relativistic quantum number (see Table 6.6)
occup	occupation number (including spin), fractial occupations supported

>>>**: line 2** is repeated for each orbital (nrorb times; see line 1)
>>>**: line 1 and 2** are repeated for each inequivalent atom. Atoms without core states (e.g. H or Li) must still include a 1s orbital, but with occupation zero.
line 3: free format

0	zero indicating end of job

7.9 MIXER (adding and mixing of charge densities)

In **mixer** the electron densities of core, semi-core, and valence states are added to yield the total new (output) density (in some calculations only one or two types will exist). Proper normalization of the densities is checked and enforced (by adding a constant charge density in the interstitial). As it is well known, simply taking the new densities leads to instabilities in the iterative SCF process. Therefore it is necessary to stabilize the SCF cycle. In **WIEN2k** this is done by mixing the output density with the (old) input density to obtain the new density to be used in the next iteration. Several mixing schemes are implemented, but we mention only:

1. straight mixing as originally proposed by Pratt (52) with a mixing factor Q
$$\rho_{new}(r) = (1-Q)\rho_{old}(r) + Q\rho_{output}(r)$$
2. a Multi-Secant mixing scheme contributed by L. Marks (see Marks and Luke 2008), in which all the expansion coefficients of the density from several preceding iterations (usually 6-10) are utilized to calculate an optimal mixing fraction for each coefficient in each iteration. It is very robust and stable (works nicely also for magnetic systems with 3d or 4f states at EF, only for ill-conditioned single-atom calculations you can break it) and usually converges at least 30 % faster than the old BROYD scheme.
3. Two new variants on the Multi-Secant method including a rank-one update (see Marks 2011) which appear to be 5-10% faster and equally robust.

At the outset of a new calculation (for any changed computational parameter such as k-mesh, matrix size, lattice constant etc.), any existing **case.broydX** files must be deleted (since the iterative history which they contain refers to a "different" incompatible calculation).

If the file **case.clmsum_old** can not be found by **mixer**, a "PRATT-mixing" with mixing factor (greed) 1.0 is done.

*Note: a **case.clmval** file must always be present, since the LM values and the K-vectors are read from this file.*

The total energy and the atomic forces are computed in mixer by reading the **case.scf** file and adding the various contributions computed in preceding steps of the last iteration. Therefore **case.scf** must not contain a certain "iteration-number" more than once and the number of iterations in the scf file must not be greater than 999.

For LDA+U calculations **case.dmatup/dn** and for hybrid-DFT (switch -eece) **case.vorbup/dn** files will be included in the mixing procedure.

With the new mode MSR1a (or MSECa) (contributed by L. Marks) atomic positions will also be mixed and thus optimized. This scheme can (unfortunately not in all cases) be a facter or 2-3 faster then the traditional optimization using **min_lapw**.

7.9.1 Execution

The program **mixer** is executed by invoking the command:

mixer mixer.def or **x mixer [-eece]**

A spin-polarized case will be detected automatically by **x** due to the presence of a case.clmvalup file. For an example see fccNi (sec. 10.2) in the **WIEN2k** package.

7.9.2 Dimensioning parameters

The following parameters are collected in file **`param.inc`**, :

NCOM	number of LM terms in density
NRAD	number of radial mesh points
NSYM	order of point group
traptouch	minimum acceptable distance between atoms in full optimization model

7.9.3 Input

Below a sample input (written automatically by **`lstart`**) is provided for TiO_2 (rutile), one of the test cases provided with the **WIEN2k** package.

```
------------------ top of file: case.inm --------------------
MSEC3 0.d0  YES  (PRATT/MSEC1/3/MSR1/a  bg charge (+1 for additional e), NORM
  0.2            MIXING GREED
1.0 1.0          Not used, retained for compatibility only
999 8            nbroyd nuse
------------------ bottom of file -----------------------
```

Interpretive comments on this file are as follows:

line 1: (A5,*)
switch, bgch, norm

switch	MSEC1	Multi-Secant scheme (Marks and Luke 2008)
	MSEC2	similar to MSEC1 (above), but mixes the higher LM values inside spheres by an adaptive PRATT scheme. This leads to a significant reduction of programsize and filesize (**`case.broyd*`**) for unitcells with many atoms and low symmetry (factor 10-50) with only slighly worse mixing performance.
	MSEC3	Similar to MSEC1, but with updated scaling, regularization and other improvements.
	MSEC4	similar to MSEC3 (above), but mixes only the L=0 LM value
	MSR1	Recommended. A Rank-One Multisecant that is slightly faster than MSEC3 in most cases. For MSR1a see later.
	MSR2	similar to MSR1 (above), but mixes only the L=0 LM value
	MSR1a	Similar to MSR1, but in addition it optimizes the atomic positions simultaneously (see Sect. 5.3.2)
	PRATT	Pratt scheme with a fixed greed
	PRAT0	Pratt scheme with a greed restrained by previous improvement, similar to MSEC3
bgch		Background charge for charged cells (+1 for additional electron, -1 for core hole, if not neutralized by additional valence electron)
norm	YES	Charge densities are normalized to sum of Z
	NO	Charge densities are not normalized

line 2: free format

greed — mixing greed Q. Essential for Pratt, rather less important for MSEC1. In the first iteration using Broyden's scheme: Q is automatically reduced by the program depending on the average charge distance :DIS andthe relative improvement in the last cycle. In case that the scf cycle fails due to large charge fluctuations, this can be further reduced but this can lead to stagnation. One should rarely reduce this below 0.05.)

line 3 (optional): (free format)
f_pw, f_clm

f_pw — Not used, retained for input compatibility.

f_clm — Not used, retained for input compatibility.

line 4 (optional): (free format)
nbroyd, nuse

nbroyd — Not used, retained for input compatibility.

nuse — For all Multisecant methods: Only nuse steps are used during modified broyden (this value has some influence on the optimal convergence. Usually 6-10 seems reasonable and 8 is recommended).

8 Programs for analysis, calculation of properties, and geometry optimization

Contents

8.1 TETRA (density of states)

This program calculates total and partial density of states (DOS) by means of the modified tetrahedron method (Blöchl et al 1994). Please note, the tetrahedron method will not work with just one k-point and **`tetra`** will automatically switch to a Gaussian broadening scheme (with default broadening of 0.01 Ry). It can also be selected by input (see below), but is not recommended for small unit cells.

It uses the partial charges in **`case.qtl`** generated by **`lapw2`** (switch QTL) and generates the DOS in states/Ry (files **`case.dos1/2/3/...`**) and in states/eV (with respect to the Fermi energy;

files **case.dos1/2/3ev**). In spin-polarized calculations the DOS is given in states/Ry/spin (or states/eV/spin).

Please note: The total DOS is equal to the sum over the atoms of the total-atomic DOS (inside spheres) and the interstitial-DOS. (Thus in the total-atomic DOS the "multiplicity" of an atom is considered). On the other hand, in the partial (lm-like) DOS the multiplicity is not considered and one obtains the total-atomic DOS as a sum over all partial DOS times the multiplicity.

The "m-decomposed" DOS (e.g. p_z, p_y, p_x) is given with respect to the local coordinate system for each atom as defined by the local rotation matrix (see Appendix A), unless you have used **x qtl** to generate the **case.qtl** and specified a specific coordinate system in **case.inq** (see Chapter 8.2).

Using the switches **-rxes E1 E2** it is possible to generate a "weight-file", where each k-point is weighted according to its contribution to the DOS in the energy range E1-E2. This weight-file **case.rxes** can be used using the switch **-rexs** to calculate the DOS with these weights. This option might be useful to simulate the E-dependency of RXES spectra, or in general calculate a "DOS" of regions around selected k-points only.

The density of states in files **case.dos1/2/3/...** or **case.dos1/2/3/...ev** can be plotted by **dosplot2_lapw** (see 5.7.4).

It is strongly recommended that you use "*Run Programs* ⇩ *Tasks* ⇩ *Density of States*" from **w2web**.

8.1.1 Execution

The program **tetra** is executed by invoking the command:

tetra tetra.def or **x tetra [-up|dn -rxes -rxesw E1 E2]**

8.1.2 Dimensioning parameters

The following parameters are listed in file **param.inc**:

MG	max. number of DOS cases
LXDOS	usually 1, except for "cross-DOS" when using TELNES.2 = 3 (not needed anymore for TELNES3)

8.1.3 Input

The required input file **case.int** can optionally be created using the **w2web**interface or the **configure_int_lapw** script (see 5.2.7).

An example is given below:

```
------------------ top of file: case.int ------------------
TiO2                          # Title
 -1.000     0.00250   1.200 0.003  # EMIN, DE, EMAX for DOS, GAUSS-Broad
    7    N   0.000                  # NUMBER OF DOS-CASES, G/L/B broadening
    0    1   tot              # jatom, doscase, description
    1    2   Ti-s
    1    3   Ti-p
    1    4   Ti-px
    1    5   Ti-py
    1    6   Ti-pz
    2    1   O-tot
------------------ bottom of file ------------------------
```

Interpretive comments on this file are as follows:

line 1: free format
title
line 2: free format
emin, delta, emax, broad

emin, delta, emax	specifies the energy mesh (in Ry) where the DOS is calculated. (emin should be set slightly below the lowest valence band; emax will be checked against the lowest energy of the highest band in **`case.qtl`**, and set to the minimum of these two values; delta is the energy increment.
broad	Gauss-broadening factor. Must be greater than delta to have any effect.

line 3: free format
ndos, Bmethod, broadening

ndos	specifies the number of DOS cases to be calculated. It should be at least 1. The corresponding output is written in groups of 7 to respective **`case.dosX`** files
Bmethod	optional input (can be omitted) to select instead of the tetrahedron method:
G	Gaussian broadening
L	Lorentzian broadening
B	both, Gauss and Lorentzian broadening
broadening	parameters in Ry, typically below/around 0.01 (optional, specify two numbers for B)

line 4: (2i5,3x,a6)
jatom, jcol, description

jatom	specifies for which atom the DOS is calculated. 0 means total DOS, $jatom = nat + 1$ means DOS in the interstitial, where nat is the number of inequivalent atoms. When spin-orbit is included, $jatom = nat + 1$ gives total spin-up/dn DOS in a spinpolarized SO calculation, but is meaningless in a non-spinpolarized SO case.
jcol	specifies the column to be used in the respective QTL-file. 1 means total, 2 . . . s, 3 . . . p, . . . The further assignment depends on the value of ISPLIT set in **`case.struct`** (see sec. 4.3); the respective description can be found in the header of **`case.qtl`**.
description	text used for further identification.

>>>**:line 4** is repeated "ndos" times

8.2 QTL (calculates special partial charges and population matrices)

This program was contributed by:

P. Novák and J.Kuneš
Inst. of Physics, Acad.Science, Prague, Czeck Republic
email: novakp@fzu.cz

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

qtl creates the input for calculating total and projected density of states of selected atoms and selected l-subshells. It thus provides similar data as **lapw2 -qtl**, but it allows for additional options. In particular it supports calculation of DOS projected on relativistic states $p_{1/2}, p_{3/2}$, $d_{3/2}, d_{5/2}$, $f_{5/2}, f_{7/2}$, DOS projected on states in a rotated coordinate system and DOS projected on individual f states. **qtl** also allows to calculate population matrix and energy resolved population matrix. Comparing to **lapwdm** population matrix, the matrix created by **qtl** may contain also the cross terms between different orbital and spin numbers and it can be energy resolved. Important option of the **qtl** is the symmetrization that makes the calculation longer, but must be switched on whenever the quantities, which are not invariant are calculated. Detailed description may be found in *QTL - technical report* by P. Novák. The calculation is based on the spectral decomposition of a density matrix on a given atomic site and its transformation to the required basis.

The output is written to **case.qtl [up/dn]**. For the DOS calculation the file **case.qtltext [up/dn]** is created in which the ordering of partial charges is given. *P*lease note, that in contrast to **case.qtl [up/dn]** from **`x lapw2 -qtl`** the total partial charge of an atom is NOT multiplied with its "multiplicity" and contains only the sum of the requested l,m terms (eg. s,p,d) and thus not all contributions. Also the interstital charge will usually be NOT correct.

8.2.1 Execution

The program **`qtl`** is executed by invoking the command:

`x qtl [ -up/dn -so -p ]` or
`qtl qtl.def`

8.2.2 Input

A sample input (a default is created automatically during **`init_lapw`** for **`case.inq`** is given below.

```
------------------ top of file: case.inq --------------------
-7.  2.          Emin Emax
 2               number of selected atoms
 1  2  0  0      iatom1  qsplit1 symmetrize loro
 2  1  2         nL1 p  d
 3  3  1  1      iatom2  qsplit2 symmetrize loro
 4  0  1  2  3   nL2 s  p  d  f
 1. 1. 1.        new axis z
 ------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

Table 8.5: Possible values of QSPLIT and their interpretation

QSPLIT	meaning
-2	DOS in basis according to ISPLIT from case.struct
-1	DOS in relativistic $\|j, l, s, m_j >$ basis
0	DOS in relativistic $\|j, l, s, m_j >$ basis, summed over m_j
1	DOS in $\|l, m_l >$ basis (no symmetry)
2	DOS in basis of real orbitals (no symmetry)
3	axial symmetry
4	hexagonal symmetry
5	cubic symmetry
6	user written unitary transformation
88	population matrix, no $< l\|l' >$ crossterms corresponds to ISPLIT=88
99	full population matrix including $< l\|l' >$ crossterms (as ISPLIT=99)

line 1: free format
emin,emax — energy window

line 2: free format
natom — number of atoms selected for calculation

line 3: free format
iatom, QSPLIT, symmetrize, loro
iatom — integer, index of atom
QSPLIT — integer, analog of ISPLIT in **case.struct**: see below
symmetrize — integer, =0 (no symmetrization), 1 (symmetrization)
loro — integer =0 original coord. system preserved
=1 (new z axis)
=2 (new z and x axes)

line 4: free format
Nl(iatom), (l(iatom,i),i=1,Nl(iatom))
Nl — number of orbital numbers selected for calculation
l — orbital numbers selected for calculation for atom iatom

line 5: free format
hz, kz, lz — real*8, direction of new axis z (if loro=1,2)

Lines starting from line 3 are repeated for each selected atom. Line 5 only appears when calculation in new coordinate system is required (loro $\neq$ 0). Axis z in this system is along hz,kz,lz (in units of the lattice vectors, need not be normalized). If not only the z axis, but also the x axis need to be specified, then loro must be equal to 2 and additional line

hx, kx, lx (real*8)

giving the direction of the new axis x, perpendicular to the new axis z must appear. For relativistic splitting (QSPLIT=0,-1) this rotation is ignored and z points along the direction of magnetization as defined in **`case.inso`**.

Indices of selected atoms, as well as the orbital numbers, must form an ascending sequence.

For QSPLIT=6 (unitary transformation prepared by user) the unitary matrices are read as in WIEN2k_07 **qtl**: For the i-th atom selected for qtl calculation, they are stored in **`case.cf$i`** and ordered according to increasing l. The unitary transformation matrix must rotate from the standard $_{lms}$-basis to the desired one. A few examples (e.g. $_{jj_z}$, $_{lms}$, or $e_g - t_{2g}$) are supplied with the code in **`$WIENROOT/SRC_templates/case.cf_*`** and must be copied to **`case.cf$i`** . For less

common cases these must be generated by hand.

8.2.3 Output

The results in file **case.qtl[up/dn]** are written in the same format as **lapw2** file **case.qtl[up/dn]** and thus they may be directly used by **tetra**.

The data for the interstital DOS correspond to $n = nat + 1$ (nat is number of atom types). The ordering of densities for all selected atoms is summarized in the file **case.qtltext[up/dn]**. The **qtltext** file that corresponds to the input data given above is:

```
Ordering of DOS in QTL file for: HoMnO3 (Munoz)

 atom   1 ordering of projected DOS
p,px,py,pz, real basis
d,dz2,d(x2-y2),dxy,dxz,dyz, real basis

 atom   3 ordering of projected DOS
 s
p,pxy,pz, axial basis
d,dz2,d(x2-y2),d(yz+xz),dxy,  axial basis
f,A2,[x(T1)+y(T1)],z(T1),[ksi(T2)+eta(T2)],zeta(T2),   axial basis
      A2=xyz  x(T1)=x(x2-3r2/5)  y(T1)=y(y2-3r2/5)  z(T1)=z(z2-3r2/5)
             ksi(T2)=x(y2-z2)   eta(T2)=y(z2-y2)   zeta(T2)=z(x2-y2)

 Data for interstital DOS correspond to atom index    8
```

The output for the population matrix integrated over energy is written to **case.dmat [up/dn]** that has the same format as analogous file calculated by **lapwdm**.

8.3 SPAGHETTI (energy bandstructure plots)

This program generates an energy bandstructure plot (postscript file **case.spaghetti_ps** and xmgrace file **case.bands.agr**) using the eigenvalues printed in **case.output1** or **case.outputso** (with switch **-so**). Using the SCF potentials one runs **x lapw1 -band** with a special k-mesh (**case.klist_band**) along some high-symmetry lines (some sample inputs can be found in **SRC_templates/*.klist** or you create your own k-mesh using **Xcrysden**). As an option, one can emphasize the character of the bands by additionally supplying corresponding partial charges (file **case.qtl** which can be obtained using **x lapw2 -qtl -band** , see 7.5). This will be called "band-character plotting" below, in which each energy is drawn by a circle whose radius is proportional to the specified character of that state. It allows to analyze the character of bands (see also figures 3.12 and 3.13).

The file **case.bands.agr** can be opened directly with **xmgrace**. Within xmgrace, all features of the plot, such as the plot range, the plot size, line properties (style, thickness and color), axis properties, labels, etc. can easily be changed by either using the menu (submenus of the "Plot" menu) or double-klicking on the corresponding part of the figure. The size of the characters for a "band-character plot" can be changed in the menu "Plot / Graph appearance / Z normalization". The figures can directly be printed or exported in eps, jpg, png and other formats, via the menus "File / Print setup" and "File / Print".

C.Persson has modified this program and it allows now also to draw connected lines. For this purpose it uses the irreducible representations (from file **case.irrep** produced by program **irrep**

together with a table of "compatibility relations" to decide which points should be connected (non-crossing rule !). *(Note: This option will NOT work on the surface of the BZ for non-symmorphic space-groups, because the corresponding group-theory has not been implemented.)*

The presence of "incompatible" **case.irrep** or **case.qtl** files (from a previous run or qtls from a DOS calculation) may crash **spaghetti**. *In such cases it is necessary to remove these files explicitly.*

It is strongly recommended that you use "*Run Programs ⇩ Tasks ⇩ Bandstructure*" from **w2web**.

8.3.1 Execution

The program **spaghetti** is executed by invoking the command:

spaghetti spaghetti.def or **x spaghetti [-up|dn] [-so] [-p]**

The -p switch directs spaghetti to use the **case.output1_*** files of a k-point parallel lapw1.

8.3.2 Input

An example is given below:

```
----------------- top of file: case.insp -------------------
### Figure configuration
 5.0   3.0                        # paper offset of plot
10.0  15.0                        # xsize,ysize [cm]
 1.0   4                          # major ticks, minor ticks
 1.0   1                          # character height, font switch
 1.1   2    4                     # line width, line switch, color switch
### Data configuration
-25.0 15.0  2                     # energy range, energy switch (1:Ry, 2:eV)
1      0.74250                    # Fermi switch,  Fermi-level (in Ry units)
1   999                           # number of bands for heavier plotting   1,1
0      1    0.02                  # jatom, jtype, size  of heavier plotting
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

line 1: free format
test

test	line must start with '###'. Begin of figure description. This tests also if you use the new input (different from WIEN97 or early WIEN2k versions)

line 2: free format
xoffset, yoffset

xoffset	x offset (in cm) of origin of plot
yoffset	y offset (in cm) of origin of plot

line 3: free format
xsize,ysize

xsize	plotsize in x direction (cm)
ysize	plotsize in y direction (cm)

line 4: free format

eincr, mtick

eincr		energy increment where y-axis labels are printed (major ticks)
mtick		number of minor ticks of y-axis

line 5: free format
charh, font

charh		scaling factor for size of labels
font	0	no text
	1	Times and Symbol
	2	Times,Times-Italic and Symbol
	3	Helvetica, Symbol, and Helvetica-Italic
	4	include your own fonts in defins.f

line 6: free format
linew, ilin, icol

linew		line width
ilin	0	dots or open circles
	1	lines
	2	lines and open circles
	3	lines and filled circles
icol	0	black
	1	one-color plot
	2	three-color plot
	3	multi-color plot
	4	multi-color plot,one color for each irred. representation

line 7: free format
test

test	line must start with '###'. Begin of data description.

line 8: free format
emin, emax, iunits

emin		energy minimum of plot
emax		energy maximum of plot
iunits		
	1	energies in Ry (internal scale)
	2	energies in eV with respect to E_f

line 9: free format
iferm, efermi

iferm	0	no line at EF
	1	solid line at EF
	2	dashed line at EF
	3	dotted line at EF
efermi		Fermi energy (Ry); can be found in the respective **`case.scf`** file. If set to 999., E_f is not plotted (and iunits=2 cannot be used)

line 10: free format

nband1, nband2	lower and upper band index for bands which should show "band-character plotting" (if **case.qtl** is present and the proper switch is set, see below). In addition the corresponding x and y coordinates are written to file **case.spaghetti_ene** (which can be used for plotting with an external xy-plotting program).

line 11: free format
jatom, jcol, jsize

jatom	If a **case.qtl** file is present, jatom indicates the atom whose character (selected by jcol) is used for "band-character plotting" (dots are replaced by circles with radii proportional to the corresponding weight, requires ilin=0,2,3). If set to zero or if **case.qtl** is not present, "band-character plotting" does not occur.
jcol	specifies the column to be used in the respective QTL-file. 1 means total, 2 . . . s, 3 . . . p, . . . The further assignment depends on the value of ISPLIT set in **case.struct**. (ignored for jatom=0). The description can be found in the header of **case.qtl**.
jsize	size factor for radii of circles used in "band-character plotting"

if line 11 is repeated, one can average the QTLs for different atoms (but with identical jcol and jsize).

8.4 IRREP (Determine irreducible representations)

This program was contributed by:

Clas Persson
Condensed Matter Theory Group,Department of Physics,
University of Uppsala, Sweden
email: Clas.Persson@fysik.uu.se

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This program determines the irreducible representation for each eigenvalue and all your k-points. It is in particular usefull to analyse energy bands and their connectivity.

You need a valid vector file, but no other input is required. The output can be found in **case.outputir** and **case.irrep**. For nonmagnetic SO calculations you must set IPR=1 in **case.inso**.

The output of this program is needed when you want to draw bandstructures with connected lines (instead of "dots").

It will not work in cases of non-symmorphic spacegroups AND k-points at the surface of the BZ. See also **$WIENROOT/SRC_irrep/README**.

8.4.1 Execution

The program **irrep** is executed by invoking the command:

irrep [up/dn]irrep.def or **x irrep [-so -up/dn]**

8.4.2 Dimensioning parameters

The following parameters are listend in file **param.inc**:

LOMAX	max. no. of local orbital. should be consistent with lapw1 and lapwso
NLOAT	number of different types of LOs
MSTP	max. step to describe k as a fraction
MAXDG	max. no. of degenerate eigenfunctions
MAXIRDG	max. no. of degenerate irr. representations
FLMAX	size of flag (FL) array (should be 4)
MAXIR	max. no. of irreducible representations
NSYM	max. no. of symmetry operations
TOLDG	min. energy deviation of degenerate states, in units of Rydberg

8.5 LAPW3 (X-ray structure factors)

This program calculates X-ray structure factors from the charge density by Fourier transformation.

You have to specify interactively valence or total charge density (because of the different normalization of **case.clmsum** and **case.clmval**) and a maximum $sin\theta/\lambda$ value.

8.5.1 Execution

The program **lapw3** is executed by invoking the command:

lapw3 lapw3.def or **lapw3c lapw3.def** or **x lapw3 [-c]**

8.5.2 Dimensioning parameters

The following parameters are listend in file **param.inc_r** or **param.inc_c** :

LMAX2	highest L in in LM expansion of charge and potential
NCOM	number of LM terms in density
NRAD	number of radial mesh points

8.6 LAPW5 (electron density plots)

This program generates the charge density (or the potential) in a specified plane of the crystal on a two dimensional grid which can be used for plotting with an external contour line program of your choice. Depending on the input files one can generate valence (**case.clmval**) or difference densities (i.e. crystalline minus superposed atomic densities) using the additional file (**case.sigma**). In spinpolarized cases one can produce up-, dn- and total densities but also spin densities (difference up-dn). It is also possible to plot total densities (**case.clmsum**), Coulomb (**case.vcoul**), exchange-correlation (**case.r2v**) or total (**case.vtotal**) potentials, but in those cases the file **lapw5.def** has to be edited and you must replace **case.clmval** by the respective filename. The file **case.rho** contains in the first line

npx, npy, xlength, ylength;

and then the density (potential) written with:

```
      write(21,11) ((charge(i,j),j=1,npy),i=1,npx)
 11   format(5e16.8)
```

It is strongly recommended that you use "*Run Programs* ⇩ *Tasks* ⇩ *Electron density plots*" from **w2web**, see the TiC example in Fig.3.6 .

8.6.1 Execution

The program **lapw5** is executed by invoking the command:

lapw5 lapw5.def or **lapw5c lapw5.def** or **x lapw5 [-c -up|dn]**

8.6.2 Dimensioning parameters

The following parameters are listend in file **param.inc**:

LMAX2	highest L in in LM expansion of charge and potential
NCOM	number of LM terms in density
NRAD	number of radial mesh points
NPT00	number of radial mesh points beyond RMT
NSYM	order of point group

8.6.3 Input

An example is given below. You may want to use XCRYSDEN by T.Kokalj to generate this file (see sect. 9.23.2).

```
---------------- top of file: case.in5 --------------------
0 0 0 1           # origin of plot: x,y,z,denominator
1 1 0 1           # x-end of plot
0 0 1 2           # y-end of plot
3 3 3             # x,y,z nshells (of unit cells)
100 100           # nx,ny
RHO               # RHO/DIFF/OVER; ADD/SUB or blank
ANG VAL NODEBUG   # ANG/ATU, VAL/TOT, DEBUG/NODEBUG
NONORTHO          # optional line: ORTHO|NONORTHO
------------------ bottom of file -----------------------
```

Interpretive comments on this file are as follows:

line 1: free format

ix,iy,iz,idv	The plane and section of the plot is specified by three points in the unit cell, an origin of the plot, an x-end and an y-end. The first line specifies the coordinates of the origin, where x=ix/idv, ... in units of the lattice vectors (except fc, bc and c lattices, where the lattice vectors of the conventional cell are used)

line 2: free format

ix,iy,iz,idv	coordinates of x-end

line 3: free format

ix,iy,iz,idv	coordinates of y-end (The two directions x and y must be orthogonal to each other unless NONORTHO is selected). Since it is quite difficult to specify those 3 points for a rhombohedral lattice, an auxiliary program **`rhomb_in5`** is provided, which creates those points when you specify 3 atomic positions which will define your plane. You can find this program using "*Run Programs* ⇩ *Other Goodies*" from **w2web**.

line 4: free format

nxsh, nysh, nzsh	specifies the number of nearest neighbor cells (in x,y,z direction) where atomic positions are generated (needs to be increased for very large plot sections, otherwise some "atoms" are not found in the plot)

line 5: free format

npx, npy	specifies number of grid points in plot. npy=1 produces a file **`case.rho_onedim`** containing the distance r (from the origin) and the respective density, which can be used in a standard x-y plotting program.

line 6: format (2a4)
switch, addsub

switch	RHO	charge (or potential) plots, no atomic density is used (regular case)
	DIFF	difference density plot (crystalline - superposed atomic densities), needs file **`case.sigma`** (which is generated with LSTART, see section 6.4)
	OVER	superposition of atomic densities, needs file **`case.sigma`**
addsub	NO	(or blank field): use only the file from unit 9
	ADD	adds densities from units 9 and 11 (if present), e.g. to add spin-up and down densities.
	SUB	subtracts density of unit 11 (if present) from that of unit 9 (e.g. for the spin-density, which is the difference between spin-up and down densities).

line 7: format (3a4)
iunits, cnorm, debug

iunits	ATU	density (potential) in atomic units e/a.u.^{3} (or Ry)
	ANG	density in e/Å^3 (do not use this option for potentials)
cnorm		determines normalization factor
	VAL	used for files **`case.clmval`**, **`r2v`**, **`vcoul`**, **`vtotal`**
	TOT	used for files **`case.clmsum`**
debug	DEBU	debugging information is printed (large output)

line 8: free format

noorth1	ORTHO	(default) enforces directions to be orthogonal
	NONORT	directions can be arbitrary; use this option only if your plotting program supports non orthogonal plots (e.g. for XCRYSDENS).

In order to plot total densities or potentials (see cnorm as above) you have to create **lapw5.def** using **x lapw5 -d**, then edit **lapw5.def** and insert proper filenames (**case.clmval**, **case.r2v**, **case.vcoul**, **case.vtotal**) for units 9 and 11, and finally run **lapw5 lapw5.def**.

8.7 AIM (atoms in molecules)

This program was contributed by:

Javier D. Fuhr and Jorge O. Sofo
Instituto Balseiro and Centro Atomico Bariloche
S. C. de Bariloche - Rio Negro, Argentina
email: fuhr@cab.cnea.gov.ar and sofo@cab.cnea.gov.ar

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This program analyses the topology of the electron density according to Bader's "Atoms in molecules" theory. For more information see Bader 2001 and Sofo and Fuhr 2001.

The original code has been significantly speeded-up by L.Marks (L-marks@northwestern.edu). There are some new optional keywords in the input (usually not needed, more for testing) and also more debugging output. All changes are described in **$WIENROOT/SRC_aim/Notes.txt**.

Basically it performs two different tasks, namely searching for "critical points" (CP) and/or determination of the atomic basins with an integration of the respective charge density.

It is important that you provide a "good" charge density, i.e. one which is well converged with respect to LMMAX in the CLM-expansion (you may have to increase the default LM-list to LM=8 or 10) and with as little "core-leakage" as possible (see **lstart**, sect. 6.4), otherwise discontinuities appear at the sphere boundary.

8.7.1 Execution

The program **aim** is executed by invoking the command:

aim aim.def or **aimc aim.def** or **x aim [-c]**

8.7.2 Dimensioning parameters

The following parameters are listed in file **param.inc**:

LMAX2	highest L in in LM expansion of charge and potential
NRAD	number of radial mesh points
NSYM	order of point group

8.7.3 Input

The input file contains "SWITCHES", followed by the necessary parameters until an END-switch has been reached.

Examples for "critical-point" searches and "charge-integration" are given below:

```
---------------- top of file: case.inaim --------------------
CRIT
1                # index of the atom (counting multiplicity)
ALL              # TWO/THRE/ALL /FOUR
3 3 3            # x,y,z nshells (of unit cells)
END
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

line 1: A4

CRIT	Keyword to calculate critical points

line 2: free format

iatom	index of the atom from where the search should be started. This count includes the multiplicity, i.e. if the first atom has MULT=2, the "second atom" has iatom=3 (Do not use simply the atom-numbers from **`case.struct`**)

line 3: A4

KEY	TWO, THRE, ALL, or FOUR defines the starting point for the CP search to be in the middle of 2, 3 or 4 atoms. ALL combines option TWO and THRE together.

line 4: free format
nxsh, nysh, nzsh

specifies the number of nearest neighbor cells (in x,y,z direction) where atomic positions are generated.

lines 1-4 can be repeated with different atoms or KEYs
line 5: A4

END	specifies end of job.

In **`case.outputaim`** the critical points are marked with a label **`:PC`**

```
:PC a1 a2 a3 l1 l2 l3 c lap rho iat1 dist1 iat2 dist2
```

where a1,a2,a3 are the coordinates of the CP in lattice vectors; l1 l2 l3 are the eigenvalues of the Hessian at the CP; c is the character of the CP (-3, -1, 1 or 3); lap is the Laplacian of the density at the CP (lap=l1+l2+l3) and rho is the density at the CP (all in atomic units). In case of a bond critical point (c=-1) also the nearest distances (dist1, dist2) to the two nearest atoms (iat1, iat2) are given.

For convenience run **extractaim_lapw case.outputaim** (see 5.2.10) and get in the file **critical_points_ang** a comprehensive list of the CP (sorted and unique) with all values given in $\AA, e/\AA^3, ...$ (instead of bohr).

```
---------------- top of file: case.inaim --------------------
SURF
3                                atom in center of surface (including MULT)
40 0.0 3.1415926536              theta, 40 points, from zero to pi
40 -0.7853981634 2.3561944902    phi
0.07 0.8 2                       step along gradient line, rmin, check
1.65 0.1                         initial R for search, step (a.u)
3 3 3                            nshell
IRHO                             "INTEGRATE" rho
WEIT                             WEIT (surface weights from case.surf), NOWEIT
30                               30 radial points outside min(RMIN,RMT)
END
------------------- bottom of file -----------------------
```

Interpretive comments on this file are as follows:

line 1: A4

SURF	Keyword to calculate the Bader surface.

line 2: free format

iatom	index of the atom from where the search should be started. This count includes the multiplicity, i.e. if the first atom has MULT=2, the "second atom" has iatom=3 (Do not use simply the atom-numbers from **case.struct**)

line 3: free format
ntheta, thmin, thmax

ntheta	number of theta directions for the surface determination. This (and nphi) determines the accuracy (and computing time).
thmin	starting angle for theta
thmax	ending angle for theta. If you have higher symmetry, you can change the angles thmin=0, thmax=π and use only the "irreducible" part, i.e. when you have a mirror plane normal to z (see case.outputs), restrict thmax to $\pi/2$.

line 4: free format
nphi, phimin, phimax

nphi	number of phi directions for the surface determination
phimin	starting angle
phimax	ending angle. (see comments for theta to reduce phi from the full $0-2\pi$ integration).

line 5: free format
h0, frmin, nstep

h0	step in real space to follow the gradient (~ 0.1)
frmin	defines the radius, for which the routine assumes that the search path has entered an atom, given as "rmin = frmin * rmt" (0.8-1.0)
nstep	number of steps between testing the position being inside or outside of the surface (2-8).

line 6: free format
r0, dr0

r0	initial radius for the search of the surface radius (1.5)
dr0	step for the search of the surface radius(0.1)

line 7: free format
nxsh, nysh, nzsh

specifies the number of nearest neighbor cells (in x,y,z direction) where atomic positions are generated.

line 8: A4

IRHO integrate function on "unit 9" (usually **`case.clmsum`**) inside previously defined surface (stored in **`case.surf`**).

line 9: A4

WEIT specifies the use of weights in **`case.surf`**.

line 9: free format

npt specifies number of points for radial integration outside the MT (30)

line 8: A4

END specifies end of job.

8.8 LAPW7 (wave functions on grids / plotting)

This program was contributed by:

Uwe Birkenheuer
Max-Planck-Institut für Physik komplexer Systeme
Nöthnitzer Str. 38, D-01187 Dresden, Germany
email: birken@mpipks-dresden.mpg.de
and
Birgit Adolph,
University of Toronto, T.O., Canada

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

The program **`lapw7`** generates wave function data on spatial grids for a given set of k-points and electronic bands. **`lapw7`** uses the wave function information stored in **`case.vector`** (or in reduced (filtered) form in **`case.vectorf`** which can be obtained from **`case.vector`** by running the program **`filtvec`**). Depending on the options set in the input file **`case.in7(c)`** one can generate the real or imaginary part of the wave functions, it's modulus (absolute value) or argument, or the complex wave function itself. For scalar-relativistic calculations both the large and

the small component of the wave functions can be generated (only one at a time). The wave functions are generated on a grid which is to be specified in the input file(s). The grid can either be any arbitrary list of points (to be specified free-formatted in a separate file **case.grid**) or any n-dimensional grid ($n = 0...3$). The operating mode and grid parameters are specified in the input file **case.in7(c)**. As output **lapw7** writes the specified wave function data for further processing – e.g. for plotting the wave functions with some graphical tools such as gnuplot – in raw format to **case.psink**. For quick inspection, a subset of this data is echoed to the standard output file **case.outputf** (the amount of data can be controlled in the input). In case, **lapw7** is called many times for one and the same wave function, program overhead can be reduced, by first storing the atomic augmentation coefficients A_{lm}, B_{lm} (and C_{lm}) to a binary file **case.abc**. For the spin-polarized case two different calculations have to be performed using either the spin-up or the spin-down wave function data as input.

It should be easy to run **lapw7** in parallel mode, and/or to apply it to wave function data obtained by a spin-orbit interaction calculation. None of these options have been implemented so far. Also, **lapw7** has not yet been adapted for **w2web**.

Please note: **lapw7** requires an LAPW basis set and does not work with APW+lo yet.

8.8.1 Execution

The program **lapw7** is executed by invoking the command:

lapw7 lapw7.def or **lapw7c lapw7.def** or **x lapw7 [-c] [-up|dn] [-sel]**

With the **-sel** option **lapw7** expects data from the reduced (filtered) wave function file **case.vectorf**, otherwise the standard wave function file **case.vector** is used. The reduced vector file **case.vectorf** is assumed to resist in the current working directory, while the standard vector file **case.vector** (which may become quite large) is looked for in the `WIEN` scratch directory. For details see **lapw7.def**.

8.8.2 Dimensioning parameters

The following parameters are listed in file **param.inc_(r/c)**:

NRAD	number of radial mesh points
NSYM	order of point group
LMAX7	maximum L value used for plane wave augmentation
LOMAX	maximum L value used for local orbitals

The meaning of `LMAX7` is the same as that of `LMAX2` in **lapw2** and that of `LMAX-1` in `lapw1`. Rather than being an upper bound it directly defines the number of augmentation functions to be used. It may be set different to `LMAX2` in **lapw2** or `LMAX-1` in **lapw1**, but it must not exceed the latter one. Note that, the degree of continuity of the wave functions across the boundary of the muffin tin sphere is quite sensitive to the choice of the parameter `LMAX7`. A value of 8 for `LMAX7` turned out to be a good compromise.

8.8.3 Input

A sample input is given below. It shows how to plot a set of wave functions on a 2-dim. grid.

```
- - - - - - - - - - - - - - - - - - - - top of file - - - - - - - - - - - - - - - - - -
2D ORTHO                # mode  O(RTHOGONAL)|N(ON-ORTHOGONAL)
0 0 0 2                 # x, y, z, divisor of origin
3 3 0 2                 # x, y, z, divisor of x-end
0 0 3 2                 # x, y, z, divisor of y-end
141 101 35 25           # grid points and echo increments
NO                      # DEP(HASING)|NO (POST-PROCESSING)
RE  ANG LARGE           # switch  ANG|ATU|AU  LARGE|SMALL
1 0                     # k-point, band index
- - - - - - - - - - - - - - - - - - - - end of file - - - - - - - - - - - - - - - - - -
```

Interpretive comments on this file are as follows.

line 1: format(A3,A1)
mode flag

mode		the type of grid to be used
	ANY	An arbitrary list of grid points is used.
	0D, 1D, 2D, or 3D	An n-dim. grid of points is used. $n = 0, 1, 2,$ or 3.
flag		orthogonality checking flag (for n-dim. grids only)
	N	The axes of the n-dim. grid are allowed to be non-orthogonal.
	O or ⟨blank⟩	The axes of the n-dim. grid have to be mutual orthogonal.

line 2: free format — (for n-dim. grids only)

ix iy iz idiv	Coordinates of origin of the grid, where x=ix/idv etc. in units of the *conventional* lattice vectors.

line 3: free format — (for n-dim. grids with $n > 0$ only)

ix iy iz idiv	Coordinates of the end points of each grid axis. This input line has to be repeated n-times.

line 4: free format — (not for 0-dim. grids)

np ... npo ...	In case of an n-dim. grid, first the number of grid points along each axis, and then the increments for the output echo for each axis. Zero increments means that only the first and last point on each axis are taken. In case of an arbitrary list of grid points, the total number of grid points and the increment for the output echo. Again a zero increments means that only the first and last grid point are taken. Hence, for n-dim. grids, altogether, $2*n$ integers must be provided; for arbitrary lists of grid points two intergers are expected.

line 5: format(A3)

tool		post-processing of the wave functions
	DEP	Each wave function is multiplied by a complex phase factor to align it (as most as possible) along the real axis (the so-called DEP(hasing) option).
	NO	No post-processing is applied to the wave functions.

line 6: format(A3,1X,A3,1X,A5)
switch iunit whpsi

switch		the type of wave function data to generate
	RE	The real part of the wave functions is evaluated.
	IM	The imaginary part of the wave functions is evaluated.
	ABS	The absolute value of the wave functions is evaluated.
	ARG	The argument the wave functions in the complex plane is evaluated.
	PSI	The complex wave functions are evaluated.
iunit		the physical units for wave function output
	ANG	Å units are used for the wave functions.
	AU or ATU	Atomic units are used for the wave functions.
whpsi		the relativistic component to be evaluated
	LARGE	The large relativistic component of wave function is evaluated.
	SMALL	The small relativistic component of wave function is evaluated.

line 7: free format
iskpt iseig

iskpt	The k-points for which wave functions are to be evaluated. Even if the wave function information is read from **`case.vectorf`**, iskpt refers to the index of the k-point in the original **`case.vector`** file! If iskpt is set to zero, all k-points in **`case.vector(f)`** are considered.
iseig	The band index for which wave functions are to be evaluated. Even if the wave function information is read from **`case.vectorf`**, iseig refers to the band index in the original **`case.vector`** file! If iseig is set to zero, all bands (for the selected k-point(s)) which can found in **`case.vector(f)`** are considered.

line 8: format(A4) — this line is optional

handle		augmentation coefficient control flag
	SAVE or STOR(E)	Augmentation coefficients are stored in **`case.abc`**). No wave function data is generated in this case. This option is only allowed if a *single* wave function is selected in the previous input line.
	READ or REPL(OT)	Previously stored augmentation coefficients are read in (from **`case.abc`**). This option is only allowed if the *same* single wave function as the one who's augmentation coefficients are stored in **`case.abc`** is selected in the previous input line.
	anything else	Augmentation coefficients are generated from the wave function information in **`case.vector(f)`**.

8.9 FILTVEC (wave function filter / reduction of case.vector)

This program was contributed by:

Uwe Birkenheuer
Max-Planck-Institut für Physik komplexer Systeme
Nöthnitzer Str. 38, D-01187 Dresden, Germany
email: birken@mpipks-dresden.mpg.de
and
Birgit Adolph
University of Toronto, T.O., Canada

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

The program **filtvec** reduces the information stored in **case.vector** files by filtering out a user-specified selection of wave functions. Either a fixed set of band indices can be selected which is used for all selected k-points (global selection mode), or the band indices can be selected individually for each selected k-point (individual selection mode). The complete wave function and band structure information for the selected k-points and bands is transferred to **case.vectorf**. The information on all other wave functions in the original file is discarded. The structure of the generated **case.vectorf** file is identical to that of the original **case.vector** file. Hence, it should be possible to use **case.vectorf** as substitutes for **case.vector** anywhere in the **WIEN** program package. (This has only been tested for **lapw7**.and **filtvec**.) To filter vector files from spin-polarized calculations, **filtvec** has to be run separately for both the spin-up and the spin-down files.

filtvec has not yet been adapted for **w2web**.

8.9.1 Execution

The program **filtvec** is executed by invoking the command:

filtvec filtvec.def or **filtvecc filtvec.def** or **x filtvec [-c] [-up|dn]**

In accordance with the file handling for **lapw1** and **lapw7** the input vector file **case.vector** is assumed to be located in the WIEN scratch directory, while the reduced output vector file **case.vectorf** is written to the current working directory. See **filtvec.def** for details.

8.9.2 Dimensioning parameters

The following parameters are listed in file **param.inc_(r/c)**:

NKPT	number of k-points
LMAX	maximum number of L values used (as in **lapw1**)
LOMAX	maximum L value used for local orbitals (as in **lapw1**)

The parameter LMAX and LOMAX must be set precisely as in **lapw1**; all other parameters must not be chosen smaller than the corresponding parameters in **lapw1**.

8.9.3 Input

Two examples are given below. The first uses global selection mode; the second individual selection mode.

I. Global Selection Mode

```
- - - - - - - - - - - - - - - - - - top of file - - - - - - - - - - - - - - - - - -
3  1 17 33     # number of k-points, k-points
2  11 -18      # number of bands, band indices
- - - - - - - - - - - - - - - - - - end of file - - - - - - - - - - - - - - - - - -
```

Interpretive comments on this file are as follows.

line 1: free format
kmax ik(1) ... ik(kmax) — Number of k-point list items, followed by the list items themselves. Positive list items mean selection of the k-point with the specified index; negative list items mean selection of a range of k-points with indices running from the previous list item to the absolute value of the current one. E.g. the sequence 2 -5 stands for 2, 3, 4, and 5.

line 2: free format
nmax ie(1) ... ie(nmax) — Number of band index items, followed by the list items themselves. Again, positive list items mean selection of a single band index; negative list items mean selection of a range of band indices.

II. Individual Selection Mode

```
- - - - - - - - - - - - - - - - - - top of file - - - - - - - - - - - - - - - - - -
2 :                    # number of k-points
17    4  11 13 15 17    # k-point, number of bands, band indices
33    3  11 -14 18      # k-point, number of bands, band indices
- - - - - - - - - - - - - - - - - - end of file - - - - - - - - - - - - - - - - - -
```

Interpretive comments on this file are as follows.

line 1:	free format kmax	the number of individual k-points to be selected. This number must be followed by any text, e.g. 'SELECTIONS' or simply ':', to indicate individual selection mode.
line 2:	free format ik nmax ie(1) ... ie(nmax)	First the index of the selected k-point, then the number of band index items, followed by the list items for the current k-point themselves. Positive list items mean selection of the band with the specified index; negative list items mean selection of a range of band indices running from the previous list item to the absolute value of the current one. E.g. the sequence 3 -7 stands for 3, 4, 5, and 7. This input line has to be repeated *kmax*-times.

8.10 XSPEC (calculation of X-ray Spectra)

This program calculates near edge structure of x-ray absorption or emission spectra according to the formalism described by Neckel et al.75, Schwarz et al. 79 and 80. For a brief introduction see below. It uses the partial charges in **`case.qtl`**. This file must be generated separately using **`lapw2`**. Partial densities of states in **`case.dos1ev`** are generated using the **`tetra`** program. Spectra are calculated for the dipole allowed transitions, generating matrix elements, which are multiplied with a radial transition probability and the partial densities of states. Unbroadened spectra are found in the file **`case.txspec`**, broadened spectra in the file **`case.xspec`**. Other generated files are: **`case.m1`** (matrix element for the selection rule L+1) and **`case.m2`** (matrix element for the selection rule L-1) and **`case.corewfx`** (radial function of the core state). The calculation is done with several individual programs (**`initxspec`**, **`tetra`**, **`txspec`**, and **`lorentz`**). which are linked together with the c-shell script **`xspec`**.

It is strongly recommended that you use "*Run Programs* ⇩ *Tasks* ⇩ *X-ray spectra*" from **w2web**.

8.10.1 Execution

Execution of the shell script xspec

The program **`xspec`** is executed by invoking the command:

`xspec xspec.def` or **`x xspec [-up|-dn]`**

Sequential execution of the programs

Besides calculating the X-ray spectra in one run using the **`xspec`** script, calculations can be done "by hand", i.e. step by step, for the sake of flexibility.

initxspec This program generates the appropriate input file **case.int**, according to the dipole selection rule, for the subsequent execution of the tetra program.
The program **initxspec** is executed by invoking the command:

initxspec xspec.def or **x initxspec [-up|-dn]**

tetra The appropriate densities of states for (L+1) and (L-1) states respectively are generated by execution of the **tetra** program.
The program **tetra** is executed by invoking the command:

tetra tetra.def or **x tetra [-up|-dn]**

txspec This program calculates energy dependent dipole matrix elements. Theoretical X-ray spectra are generated using the partial densities of states (in the **case.dos1ev** file) and multiplying them with the corresponding dipole matrix elements.
The program **txspec** is executed by invoking the command:

txspec xspec.def or **x txspec [-up|-dn]**

lorentz The calculated spectra must be convoluted to account for lifetime broadening and for a finite resolution of the spectrometer before they can be compared with experimental spectra. In the **lorentz** program a Lorentzian is used to achieve this broadening.
The program **lorentz** is executed by invoking the command:

lorentz xspec.def or **x lorentz [-up|-dn]**

8.10.2 Dimensioning parameters

The following dimensioning parameters are collected in the files **param.inc** of **SRC_txspec** and **SRC_lorentz**:

IEMAX0	maximum number of energy steps in the spectrum (SRC_lorentz)
NRAD	number of radial mesh points
LMAX	highest l+1 in basis function inside sphere (consistent with input in case.in1)

8.10.3 Input

Two examples are given below; one for emission spectra and one for absorption spectra:

Input for Emission Spectra:

```
---------------- top of file: case.inxs --------------------
NbC: C K                (Title)
2               (number of inequivalent atom)
1               (n core)
0               (l core)
0,0.5,0.5       (split, int1, int2)
-20,0.1,3       (EMIN,DE,EMAX   in eV)
EMIS            (type of spectrum, EMIS or ABS)
0.35            (S)
0.25            (gamma0)
0.3             (W)
AUTO            (generate band ranges AUTOmatically or MANually
-7.21           (E0 in eV)
-10.04          (E1 in eV)
-13.37          (E2 in eV)
------------------- bottom of file ------------------------
```

Input for Absorption Spectra:

```
---------------- top of file: case.inxs --------------------
NbC: C K        (Title)
```

```
2               (number of inequivalent atom)
1               (n core)
0               (l core)
0,0.5,0.5       (split, int1, int2)
-2,0.1,30       (EMIN,DE,EMAX   in eV)
ABS             (type of spectrum)
0.5             (S)
0.25            (gamma0)
------------------ bottom of file -----------------------
```

Interpretive comments on these files are as follows.

line 1: free format

TITLE	Title

line 2: free format

NATO	Number of the selected atom (in case.struct file)

line 3: free format

NC	principle quantum number of the core state

line 4: free format

LC	azimuthal quantum number of the core state

The table below lists the most commonly used spectra:

Spectrum	n	l
K	1	0
$L_{II,III}$	2	1
M_V	3	2

Table 8.50: Quantum numbers of the core state involved in the x-ray spectra

line 5 free format

SPLIT, INT1, INT2	split in eV between e.g. L_{II} and L_{III} spectrum (compare with the respective core eigenvalues), INT1 and INT2 specifies the relative intensity between these spectra. Values of 0, 0.5, 0.5 give unshifted spectra.

line 6: free format

EMIN, DE, EMAX	minimum energy, energy increment for spectrum, maximum energy; all energies are in eV and with respect to the Fermi level EMIN and EMAX are only used as limits if the energy range created by the **`lapw2`** calculation (using the QTL switch) is greater than the selected range.

line 7: Format A4

TYPE	EMIS	X-ray emission spectrum
	ABS	X-ray absorption spectrum (default)

line 8: free format

S	broadening parameter for the spectrometer broadening. For absorption spectra S includes both experimental and core broadening. Set S to zero for no broadening.

line 9: free format

GAMMA0	broadening parameter for the life-time broadening of the core states. Set GAMMA0 to zero to avoid lifetime broadening of the core states.

line 10: free format

W	broadening parameter for the life-time broadening of valence states. Set W to zero to avoid lifetime broadening of the valence states.

line 11: format A4

BANDRA

AUTO	band ranges are determined AUTOmatically (default)
MAN	band ranges have to be entered MANually

line 12: free format

E0	Emission spectra: onset energy for broadening, E0, generated automatically if AUTO was set in line 10 Absorption spectra: not used

line 13: free format

E1	Emission spectra: onset energy for broadening, E1, generated automatically if AUTO was set in line 10 Absorption spectra: not used

line 14: free format

E2	Emission spectra: onset energy for broadening, E2, generated automatically if AUTO was set in line 10 Absorption spectra: not used

8.11 TELNES3 (calculation of energy loss near edge structure)

This program was contributed by:

Kevin Jorissen and Cécile Hébert
Ecole Polytechnique Federale de Lausanne

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

The TELNES3 program calculates the double differential scattering cross section (DDSCS) on a grid of energy loss values and impulse transfer vectors. This double differential cross section is integrated to yield a differential cross section, which is written to file. The differential cross section is either a function of energy (ELNES integrated over impulse transfer q); or a function of impulse transfer (ELNES integrated over energy loss E), which shows the angular behavior of scattering.

The DDSCS is calculated as described in a forthcoming publication by K. Jorissen, C. Hebert, and J. Luitz. (The Ph.D. thesis of K. Jorissen (http://www.wien2k.at/reg_user/textbooks/) also describes the formalism onto which TELNES3 is built in great detail.) This formalism allows calculation of relativistic EELS including transitions of arbitrary order (i.e., non-dipole transitions). It takes into account the relative orientation between sample and beam. If this is not necessary (because the crystal is isotropic, or the sample is polycrystalline), the formula may be integrated over 4π, simplifying the calculation. Both scenarios are implemented in TELNES3.

A note to our faithful fans from the early days: it used to be necessary to play such tricks as recompiling **lapw2** with lxdos=3 ; to create k-meshes without symmetry ; and to edit case.struct and set ISPLIT to 99. This is no longer necessary. Just sit back, relax, and press the buttons in w2web. The integration with the package **qtl** will do the job.

8.11.1 Execution

Execution

The program **telnes3** is executed by invoking the command:

telnes3 telnes3.def or **x telnes3 [-up|-dn]**

8.11.2 Input

TELNES3 requires one input file - **case.innes**. We recommend using **InnesGen**^{TM} of **w2web** to create this input file in a clear and intuitive way. If you wish to manually edit the file, please refer to the following description. Please note that input files created for TELNES2 may or may not work with TELNES3, depending on which optional keywords were used. There isn't a shred of compatibility with the old TELNES program.

The file **case.innes** consists of two parts: a first block with required input, and a second block with optional input. In fact, the second part may be omitted altogether. The simplest input file looks like this:

```
Graphite C K edge of first atom.
1            (atom)
1, 0         (n, l core)
285          (E-Loss of 1st edge in eV)
300          (energy of the incident electrons in keV)
0.0 20.0 0.1 (the energy mesh)
5.0  1.87    (collection semiangle, convergence semiangle, both in mrad)
10 1         (NR, NT, defining the integration mesh in the detector plane)
0.8          (spectrometer broadening in eV)
END
```

This first part of the file is not formatted and contains the following information:

line	value	explanation
1	'Graphite ...'	Title (of no consequence for the calculation)
2	1	Atom number as given in case.struct (the index which numbers inequivalent atoms)
3	1 0	main and orbital quantum number n and l of the core state; eg. 1 0 stands for 1 s
4	285	energy of the edge onset in eV (here for the C K edge)
5	300	beam energy in keV
6	0.0 20.0 0.1	energy mesh given as $E_{min}E_{max}E_{step}$; all values in eV. 0.0 is the edge threshold.
7	5.0 1.87	detector collection semiangle and microscope convergence semiangle in mrad
8	10 1	parameters NR and NT which determine the mesh used for sampling the distribution of Q-vectors allowed by collection and convergence angles
9	0.8	spectrometer broadening FWHM in eV
10	END	keyword telling the program that there is no more input to read. **Optional keywords and values must be inserted before this line!**

There are many other parameters that control the calculation, most of which are set to reasonable default values. To use these advanced parameters, add corresponding keywords **before** the END keyword. We recommend using **InnesGen^{TM}** of **w2web** to create this input file.

Currently, the keywords listed below may be used. Although only the first four characters of each keyword are read, we recommend using the full keyword for clarity.

```
VERBOSITY
n                eg. : 1
```

Specifies how much output you'll get. n must be 0 (only basic output; default), 1 (medium output) or 2 (full output, including more technical information).

```
ATOMS
n1 n2             eg. : 1 3    (default : 1 0 == 1 mult(natom) )
```

The atom number on line 2 (see above) corresponds to a class of equivalent atoms in case.struct. Equivalent positions n1 to n2 will contribute to the spectrum (default : sum over all atoms in the equivalency class). Since all equivalent atoms have identical electronic structure up to a symmetry operation, this will simply yield a prefactor (n2-n1+1) for the orientation averaged spectrum, but as each equivalent atom has a different orientation with respect to the beam, this setting will influence the shape of an orientation sensitive spectrum.

```
DETECTOR POSITION
theta_x theta_y            eg. : 0.5 0.5   (default : 0 0)
```

By default, the detector is aligned with the incoming beam - i.e., source, sample, and detector are connected by a straight line. This card shifts the detector in a plane perpendicular to the incoming beam. The shift is expressed as an angle in mrad. If one draws a line between source and sample, and another line from the sample to the center of the detector aperture, these 2 lines will form an angle of $\sqrt{theta_x^2 + theta_y^2}$ mrad.

```
MODUS
m                 eg. : angles  (default is energy)
```

The output is a spectrum as a function of energy if m=energy. The output is a spectrum as a function of impulse transfer/scattering angle if m=angle.

```
SPLIT
splitting energy eg. : 2.7
```

If the initial state has an orbital quantum number larger than 0, it will generate two superposed edges: one corresponding to $j = l - 1/2$, and one corresponding to $j = l + 1/2$ (eg., for the 2p initial state we have a L3 and a L2 edge). The splitting energy sets the energy separation of the two edges and should be given in eV (here, L3 is at the energy specified in the beginning of case.innes, and L2 is 2.7 eV higher). By default (keyword omitted), the splitting energy is calculated by the program. It is generally quite accurate.

```
BRANCHING RATIO
branching ratio  eg. : 1.4
```

The branching ratio is a scaling factor (eg., here the ratio of intensities L3/L2 would be set to 1.4). By default (keyword omitted), the branching ratio is set to its statistical value of $(2l + 2)/2l$.

```
NONRELATIVISTIC
```

This key tells the program not to use the relativistic corrections to the scattering cross section. This option generates spectra identical to output of the old TELNES program. This produces incorrect results in many cases. By default, relativistic calculations are done.

```
INITIALIZATION
make_dos      write_dos      eg. N N (default : Y Y)
make_rot.mat. write_rot.mat  eg. Y N (default : Y Y)
```

TELNES3 needs many ingredients for its calculations, and this key defines how it gets two of them: the density of states, and the rotation matrices (used for transforming q-vectors from one atom to an equivalent atom). The first entry says whether or not the ingredient has to be calculated (Y : calculate; N : read from file), and the second entry says whether or not the ingredient has to be written to file (Y : write; N : don't write). If make_dos=Y, a file case.qtl must be present from which the dos will be calculated. If make_dos=N, then either a file **`case.dos`** or a file **`case.xdos`** containing the (x)dos must exist. If make_rot.mat=N, a file **`case.rotij`** containg the rotation matrices must exist. If write_rot.mat=Y, a file **`case.rotij`** is written. If write_dos=Y, a file **`case.dos`** or **`case.xdos`** is written. The calculation of the rotation matrices is computationally negligible, but it is recommended to write the xdos to file and not calculate it over and over again.

```
QGRID
qmodus           eg. L    (U by default)
theta_0               eg. 0.05 (no default value)  )
```

A collection angle α and convergence angle β allow scattering angles up to $\alpha + \beta$ and a corresponding set of Q-vectors. This set (a disk of radius $\alpha + \beta$) is sampled using a discrete mesh. Three types of meshes are implemented :

U a uniform grid, where each Q-vector samples an equally large part of the disk. Sampling is set up by drawing NR equidistant circles inside the big circle, and choosing $(2i - 1)NT$ points on circle i, giving $NR^2 * NT$ points in total.

L a logarithmic grid with NR circles. The distance between circles increases exponentially. There are $(2i-1)NT$ points on circle i, and NR^2NT points in total. Circle i is at radius theta_0 $e^{((i-1)dx)}$, where dx depends on NR, α and β.

1 a one dimensional logarithmic mesh; there are NR circles at exponential positions, and only one point on each circle (so NR points in total). This means we sample a line in the detector*beam plane. An economic way of getting spectra as a function of scattering angle in cases with symmetric scattering.

The line specifying theta_0 is to be omitted for the U grid.

```
ORIENTATION SENSITIVE
g1 g2 g3        (eg. 0.0  40.0  0.0)  (no default value)
```

This key tells the program not to average over sample to beam orientations, but to use the particular sample to beam orientation defined by the three Euler angles (to be given in degrees). If the ORIENTATION SENSITIVE key is not set, the program will average over all orientations (default).

```
SELECTION RULE
type            (eg. : q)  (default : n)
```

The formula for the DDSCS contains an exponential factor in q, which we expand using the Rayleigh expansion. We identify each term in the expansion by the order lambda of the spherical Bessel function $j_\lambda(q)$ it contains. This key keeps some terms and discards others. This can be useful to eliminate unwanted transitions ; to study a spectrum in greater detail ; or simply to speed up the calculation significantly. Possible settings for 'type' are :

```
m   :  use lambda = 0 only
d   :  use lambda = 1 only
q   :  use lambda = 2 only
o   :  use lambda = 3 only
n   :  no selection rule, calculate all transitions
0-3 :  all transitions up to lambda (eg., 1 means lambda = 0 and 1)
```

Be aware that the availability of the DOS limits the possible transitions (WIEN2k gives us the DOS only up to l=3). In the nonrelativistic limit, the SELECTION RULE and LSELECTION RULE coincide i.e., the $\lambda = 1$ terms correspond to dipole transitions etc. This is no longer true in the relativistic case.

```
LSELECTION RULE
type            (eg. : q)  (default : d)
```

Whereas the previous key selects transitions by the order of the interaction potential, this key selects them by the L-character of the final states. Possible settings for 'type' are (the orbital momentum of the initial state being denoted with l):

```
m   :  L=l
d   :  L=l +/- 1
q   :  L=l +/- 2
o   :  L=l +/- 3
n   :  no selection rule, calculate all transitions
0-3 :  |L-l| <= type
```

Be aware that the availability of the DOS limits the possible transitions (WIEN2k gives us the DOS only up to l=3). In the nonrelativistic limit, the SELECTION RULE and LSELECTION RULE coincide i.e., the $\lambda = 1$ terms correspond to dipole transitions etc. This is no longer true in the relativistic case.

```
EXTEND POTENTIALS
Rmax  sampling  lmax  refine   (e.g.:  3.0  15  0  1.0)  (no defaults)
```

Calculate matrix elements beyond the muffin tin radius up to r = rmax (in Bohr units). Refine the radial grid by a factor refine (1 means default sampling density). This is done by evaluating the potential as given in case.vtotal, which must be present for this type of calculation, and reexpanding it in spherical harmonics, using an angular grid with step of sampling degrees, and expanding up to l=lmax. Currently, users should keep lmax to 0 and almost certainly refine to 1.0 . However, advanced users can play around with the software and tweak it to do interesting things if they wish. TELNES3 only requires the spherical potential l=0.

```
FERMI ENERGY
Ef     (e.g. 0.75)
```

Manually set the Fermi energy to Ef (needs to be given in Rydberg units). (The default behavior is to get Ef from the header of case.qtl.)

```
CORE WAVEFUNCTION
filename     (e.g.  case.cwf)
```

Read the wave function of the initial state from file. (Default behavior is to calculate it instead.)

```
FINAL STATE WAVEFUNCTION
filename     (e.g.  case.finalwf)
```

Read the radial wave functions of the final state from file. (Default behavior is to calculate it instead.)

```
RELATIVISTIC
Itype   (e.g. 1)
```

Determines which flavor of relativity to use : 0 means nonrelativistic (as in TELNES), 1 means fully relativistic (default), 2 means using the contracted q-vector (only valid for dipole transitions ; as in TELNES2).

```
NOHEADERS
```

Don't put headers in output files. This can be helpful if your plotting program doesn't like the headers. (Gnuplot doesn't mind them.)

```
DOSONLY
```

Don't calculate the EELS spectrum halt the program after the calculation of the density of states is finished.

```
NBTOT
nb    (e.g.    200 )
```

Arrays for the DOS are first allocated at some initial size, and then reallocated at larger size if necessary. Unfortunately, these reallocation routines appear unstable in some circumstances. This card allows the user to set an array size manually and avoid the need to reallocate (nb is the number of bands). However, very large systems may lead the system to run out of memory and cause a crash.

The following cards are not yet activated (placeholders): TABULATE, SPIN

The following cards are no longer active and must be removed or renamed: XQTL, WRONG.

8.11.3 Practical considerations

A typical ELNES calculation consists of the following steps:

- initialize (**init_lapw**) and converge a SCF calculation (**run_lapw**)
- provide a suitable **case.innes** file
- if more excited states are needed than given by the SCF calculation, raise the upper energy limit in **case.in1** and run **x lapw1**
- create the **case.qtl** file using **x qtl -telnes**
- calculate the EELS spectrum using **x telnes3**. It is generally a smart move to make the program calculate the DOS on the biggest energy grid you wiill ever need, save this to file, and simply read it from file for all future calculations (INITIALIZATION key). The same should be done for calculations using EXTEND POTENTIAL (use CORE WAVEFUNCTION key to save to file). This saves time. (In case of disk space problems, once the **case.qtl** file has been created, the **case.vector** files can be deleted. Similar, the **case.qtl** file can be deleted or compressed once the **case.dos** file exists.)
- add broadening to the spectrum using **x broadening**. If you wish, editing the **case.inb** file allows tweaking of the broadening.
- study the output (**case.elnes** or **case.broadspec** are the place to start).
- if you wish to do more calculations, save the current results using **save_eels -d calculation1** . Edit **case.innes** and run **x telnes3** again.

This sequence can conveniently be executed using **w2web** by simply clicking one button after the other.

8.11.4 Files

TELNES3 uses a lot of files. Many output files are only written if VERBOSITY is set to a high level. Many input files are required only for certain input settings in **case.innes**. We list here all files possibly used by TELNES3 (and listed in **telnes3.def**). Each filename is followed by I or O (input/output), a short description of the file content, and a comment on when the file is used.

- case.innes (I). Defines the ELNES calculation. Always read.
- case.struct (I). Defines the crystal. Always read.
- case.vsp (I). Spherical component of the crystal potential. Read unless core and final state wavefunctions are read from file.
- case.vtotal (I). Total crystal potential (can be generated by **lapw0**). Read if EXTEND POTENTIAL is used.
- case.rotij (I). Rotation matrices that transform q-vectors between equivalent atoms. Read if INITIALIZATION tells the program to do so.
- case.dos (IO). l-resolved density of states. Read or written depending on INITIALIZATION settings.

- case.xdos (IO). lm,l'm'-resolved density of states. Read or written depending on INITIALIZATION settings; only if the calculation is orientation resolved.
- case.qtl (I). contains partial charge components and Fermi energy. Read if DOS needs to be calculated (INITIALIZATION) or if Fermi energy is not specified using FERMI.
- case.inc (I). Specifies core states. Only read if core states are calculated.
- case.kgen (I). contains k-mesh to sample the Brillouin Zone. Read if DOS needs to be calculated.
- case.outputelnes (O). Main log file. Always written. Content depends on VERBOSITY.
- case.elnes (O). Total spectrum. Always written.
- case.sdlm (O). Partial (l,m) spectra. Written if verbosity > 0.
- case.ctr (O). (l,m,l'm') crossterms. Written if verbosity > 0 and calculation is orientation sensitive.
- case.corewavef (O). Contains core wavefunctions. Written if core wavefunctions were calculated and verbosity > 1.
- case.final (O). Contains APW radial basis functions for final states at selected energies. Written if verbosity > 1.
- case.ortho (O). Contains scalar products of initial and final states. Written if verbosity > 1.
- case.matrix (O). Proportionality between partial DOS and spectrum for each l-value. Written if verbosity > 0 and MODUS is energy.
- case.cdos (O). Selected (l,m,l'm') cross-DOS terms. Written if calculation is orientation sensitive and verbosity > 1 or INITIALIZATION causes DOS to be written to file.
- case.sp2 (O). Integrated cross sections as a function of collection angle for all l-values. Written if calculation is orientation sensitive, MODUS is set to angle and verbosity > 1.
- case.angular (O). Differential cross section as a function of scattering angle for all l-values. Written if calculation is orientation sensitive, MODUS is set to angle and verbosity > 1.
- case.inb (O). Settings for the broadening program. Always written.
- case.eelstable (O). Placeholder. Not currently used.
- telnes3.def (I). List of files used by TELNES3. Always read.
- telnes3.error (O). Error file containing current error message; empty after successful calculation. Always written.

8.12 BROADENING (apply broadening to calculated spectra)

This program was contributed by:

Joachim Luitz
IAST Austria
wien2k@luitz.at

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

The broadening program can be used in conjunction with the **TELNES3** or the **xspec** program to broaden theoretical spectra by applying a lorentzian broadening for core and valence life times and a gaussian broadening for spectrometer broadening.

8.12.1 Execution

Execution

The program **broadening** is executed by invoking the command:

broadening broadening.def or **x broadening**

8.12.2 Input

broadening needs one input file - **case.inb**. When running **TELNES3** this input file is automatically created from settings given in **case.innes**.

```
GaN
ELNES
1   1   0
0.0   1.0    0.0
0.116  0.116
1   2.15000000000000
0.6
dummy
0.0
0.0
0.0
```

line	value	explanation
1	'GaN ...'	Title (of no consequence for the calculation)
2	ELNES — ABS — EMIS	Type of input spectrum
3	NC C1 C2	specification of input file: NC number of columns to read, C1 and C2 column to broaden (only in "ELNES" mode)
4	SPLIT XINT1 XINT2	split energy, XINT1—2 relative intensities of spectra in C1 and C2
5	GA GB	core hole lifetime of the two edges
6	W WSHIFT	W: type of valence broadening (1: linear with $E/10$, 2: Muller like E^2), edge offset
7	S	Spectrometer broadening FWHM in eV
8	dummy	dummy keyword for compatibility with **lorentz**
9-11	E0, E1, E2	quadratic energy dependent broadening (only used for type ELNES and EMIS when selecting valence broadening type W=2)

8.13 OPTIMIZE (Volume, c/a or 2-4 dimensional lattice parameter optimization)

This program generates a series of new struct files corresponding to different volumes, c/a ratios, or otherwise different lattice parameters (depending on your input choice) from an existing struct file (either **case_initial.struct** or **case.struct**). (When **case_initial.struct** is not present, it will be generated from the original **case.struct**.

Furthermore it produces a shell script **optimize.job**. You may modify this script and execute it. Further analysis of the results (at present only equilibrium volume or c/a ratio are supported in w2web) allows to find the corresponding equillibrium parameters (see Sec.5.3.1).

8.13.1 Execution

The program **optimize** is executed by invoking the command:

optimize optimize.def or **x optimize**

8.13.2 Input

You have to specify interactively which task should be performed (volume, c/a, b/a optimization, or full optimization for tetragonal, orthorhombic or monoclinic structure), how many cases you want to do and how large the change (+/- xx %) should be for each case.

8.14 ELAST (Elastic constants for cubic cases)

This program was contributed by:

Thomas Charpin
Lab. Geomateriaux de l'IPGP, Paris, France
(In September 2001 we received the sad notice that Thomas Charpin died in a car accident).
Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This package calculates elastic constants for cubic crystals. It is described in detail by the author in Charpin 2001. Please note, at **http://www.wien2k.at/reg_user/unsupported** you can find a package **Hex-elastic** by Morteza Jamal, which should calculate elastic constants also for hexagonal symmetry.

8.14.1 Execution

The package is driven by three scripts:

- **init_elast**:
  It prepares the whole calculation and should be run in a directory with a valid **case.struct** and **case.inst** file. It creates the necessary subdirectories **elast**, **elast/eos**, **elast/tetra**, **elast/rhomb**, **elast/result**, the templates for tetragonal and rhombohedral distortion and initializes the calculations using **init_lapw**.
- **elast_setup**:
  It should be run in the **elast** directory, generates the distorted struct-files and **eos.job**, **rhomb.job** and **tetra.job**. These scripts must be adapted to your needs (spin-polarization, convergence,...) and run. **elast_setup** can be run several times (for different distortions,...).
- **ana_elast**:
  Once all calculations are done, change into **elastresult** and run this script. The final results are stored in **elastresultoutputs**.
- **genetempl, setelast, anaelast**:
  These three small programs are called by the above scripts.

8.15 MINI (Geometry minimization)

This program is usually called from the script **min_lapw** and performs movements of the atomic positions according to the calculated forces (please read Sec. 5.3.2). It generates a new **case.struct** file which can be used in the next geometry/time step. Depending on the input options, **mini** helps to find the equilibrium positions of the atoms or performs a molecular dynamics simulation (which might take very long time).

For finding the equilibrium positions different methods are available. We recommend PORT, a "reverse-communication trust-region Quasi-Newton method" from the Port library (http://www.bell-labs.com/project/PORT/doc/port3doc.tar.gz, Gay 1983), which was implemented by L.D.Marks (L-marks@northwestern.edu, http://www.numis.northwestern.edu). It minimizes the total energy and NOT the forces (using the forces as derivative of E vs. atomic positions). In cases when energy and forces are not "compatible", eg. because of numerical noise due to limited scf convergence, small RKmax or crude k-mesh, PORT may fail. An interesting alternative is a sophisticated modified steepest-descent method (NEW1), which minimizes the forces (does not use the total energy). Eventually a damped Newton dynamics is also available.

The forces are read from a file **case.finM**, while the "history" of the geometry optimization or MD is stored in **case.tmpM**

One can constrain individual positions in **case.inM** or define linear constrains for several positions using **case.constraint** (thanks to B.Yanchitsky (Kiev, yan@imag.kiev.ua); for details see comments in the SRC_templates/case.constraint file). In case of calculations with linear constrains one should use NEW1 (in **case.inM**). When constraining individual positions and using PORT, one should after modifications in **case.inM** rerun **x pairhess -copy** (which copies **.minpair** to **.minrestart** and **.min_hess**).

8.15.1 Execution

The program **mini** is executed by invoking the command:

mini mini.def or **x mini**

8.15.2 Dimensioning parameters

The following dimensioning parameters are collected in the file **param.inc**:

MAXIT	maximum number of geometry steps
NRAD	number of radial mesh points
NCOM	number of LM terms in density
NNN	number of neighboring atoms for nn
NSYM	order of pointgroup

8.15.3 Input

Two examples are given below; one for a PORT geometry optimization, and one for molecular dynamics using a NOSE thermostat:

Input for geometry optimization:

```
--------------- top of file: xxx.inM -------------------
```

```
PORT 2.0  0.25          (PORT/NEWT   tolf step0 (a4,2f5.2))
1.0 1.0 1.0 3.0         ( 1..3:delta, 4:BO/eta(1=friction zero))
1.0 1.0 1.0 6.0         ( 1..3=0 constraint)
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows.

line 1: format(a4,2f5.2)

MINMOD		Modus of the calculation
	PORT	Geometry optimization with reverse-communication trust-region Quasi-Newton routine from the Port library. Recommended option.
	NEW1	Performs geometry optimization with "sophisticated" steepest-descent method with automatic adaptation of stepsize (still experimental, but when PORT fails, an interesting alternative)
	NEWT	Performs geometry optimization with damped Newton scheme according to $R_m^{\tau+1} = R_m^{\tau} + \eta_m(R_m^{\tau} - R_m^{\tau-1}) + \delta_m F_m^{\tau}$ where R_m^{τ} and F_m^{τ} are the coordinate and force at time step τ. When the force has changed its direction from the last to the present timestep (or is within the tolerance TOLF), η_m will be set to $1 - \eta_m$. Please see also the comments in Sect. 5.3.2
	BFGS	Performs geometry optimization with the variable metric method of BFGS. This option works only when a quadratic approximation is a good approximation to the specific potential surface. Obsolete.
TOLF		Force tolerance, geometry optimization will stop when all forces are below TOLF.
STEP0		Initial "Trust-region radius". Determines size of first geometry step.

line 2: free format

DELTA(1-3)	For PORT (and BFGS): Precondition parameters: rescales the gradient and thus determines the size of the geometry steps For NEWT/NEW1: x,y,z-delta parameters. Determines speed of motion. Good values must be found for each individual system. They depend on the atomic mass, the vibrational frequencies and the starting point (see Sect. 5.3.2). DELTA(i) = 0 constrains the corresponding i-th coordinate (for PORT: after setting a DELTA(i)=0, also rerun pairhess to set a proper Hessian). The delta-x,y,z correspond to the global coordinates (the same as the positions in **`case.struct`** and the forces **`:FGL`** from **`case.scf`**). *Whenever you change these DELTA(i) you must remove file* ***`case.tmpM`****!*
ETA	For NEWT: damping (friction) parameter. ETA=1 means no friction, ETA=0 means no speed from previous time steps PORT: changes the strength of the bonds when running pairhess and ZWEIGHT is negative (see the pairhess description), otherwise not used NEW1: ETA is not used

>>> **line 2:** must be repeated for every atom

Input for Molecular dynamics:

```
---------------- top of file: nbc.inM --------------------
```

```
NOSE                      (NOSE/MOLD (a4))
58.9332 400. 1273. 5.0    (Masse, delta t, T, nose-frequency)
58.9332 400. 1273. 5.0
58.9332 400. 1273. 5.0
58.9332 400. 1273. 5.0
58.9332 400. 1273. 5.0
58.9332 400. 1273. 5.0
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows.

line 1: format(a4,f5.2)

MINMOD	Modus of the calculation
MOLD	Performs next molecular dynamics timestep
NOSE	Performs next molecular dynamics timestep using a NOSE thermostat

line 2: free format

MASS	Atomic mass of i^{th} atom
TIMESTEP	Time step of MD (in atomic units, depends on highest vibrational frequencies)
TEMP	Simulation Temperature (K)
NOSF	Nose-frequency

>>>**line 2:** must be repeated for every atom

8.16 OPTIC (calculating optical properties)

This program was contributed by:

Claudia Ambrosch-Draxl
Atomistic Modelling and Design of Materials
University Leoben
A-8700 Leoben, AUSTRIA
email: cad@unileoben.ac.at

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

The theoretical background is described in detail in Ref. Abt 1994 and Ambrosch-Draxl 06 (Please cite the latter when publishing optics results!). The calculation of optical properties requires a dense mesh of eigenvalues and the corresponding eigenvectors. For that purpose start **kgen** and generate a fine k-mesh (with many k-points). Run **lapw1** and then **lapw2** with the option FERMI (*Note: You must also put TETRA / with value=101. for metallic systems* ***case.in2***) in order to generate the weight-file. After the vector-file has been generated by **lapw1** run **optic** in order to produce the momentum matrix elements. Then the program **joint** carries out the BZ integration and computes the imaginary part of the complex dielectric tensor. In order to obtain the real part of the dielectric tensor **kram** may be executed which uses the Kramers-Kronig relations.

The program **optic** generates the symmetrized squared momentum matrix elements

$\mathbf{M}_i = < n'\vec{k}|\vec{p}.\vec{e}_i|\ n\vec{k} >^2$

between all band combinations for each k-point given in the vector-file and stores them in **case.symmat**. For the orthogonal lattices the squared diagonal components can be found in the file **case.mat_diag**. For non-orthogonal systems all 6 components $(M_j)^* M_k$ can be calculated according to the symmetry of the crystal. In systems without inversion symmetry the complex version **opticc** must be executed.

The matrix elements (and the imaginary part of the dielectric tensor) are given per spin in case of the spin-polarized calculation and as a sum of both spin directions if the calculation is non-spinpolarized.

Due to spin-orbit coupling imaginary parts of the nondiagonal elements may occur in spinpolarized cases. Thus in general, up to 9 components can be calculated at the same time.

Since version WIEN2k_11.1 an option for the calculation of **XMCD** (X-ray magnetic circular dichroism) has been added by Lorenzo Pardini (loren.pard@gmail.com). Please cite Pardini et al. 2011 when using XMCD and check the paper for further details. In the case of the XMCD calculation, the momentum matrix elements in the dipole approximation between the selected core state and conduction states are stored in case.symmat1up (higher energy core state, eg. L_3) and case.symmat2up (lower energy core state, eg. L_1) for each k-point and every band. For K, L_1, and M_1 edges, only **case.symmat1up** is written, since in these cases there is only one edge, whereas both **case.symmat1up** and **case.symmat2up** are written for the remaining cases.
XMCD calculation can be only performed for system with spin-polarized AND spin-orbit set up.
In order to calculate XMCD and x-ray absorption spectra, eigenvalues must be evaluated over a mesh in the whole Brillouin zone; for that porpouse, the following procedure should be followed:

- copy **case.struct** to **case.ksym (cp case.struct case.ksym)** and remove all the symmetry operations but the identity;
- generate a k-mesh in the whole Brilouin zone **(x kgen -so)**;
- change TOT to FERMI in **case.in2c**;
- set IPRINT=1 in **case.inc** to activate core-wavefunction output;
- for metallic systems, put *TETRA* with value 101;
- execute **runsp_lapw -so -s lapw1 -e lcore**;
- run optic: **x optic -c -so -up**;
- run joint: **x joint -up**.

*You must not use p-1/2 "relativistic" LOs in **LAPWSO**, since this basis is not supported on **OPTICS** yet.*

8.16.1 Execution

The program **optic** is executed by invoking the command:

optic(c) optic.def or **x optic [-c -up|dn -so -p]**

Recommended procedure for spin-orbit coupling:

In order to get the correct matrix elements, the files **case.vectorso[up|dn]** have to be used. For that purpose the following procedure is recommended:

- run SCF cycle: **run[sp]_lapw -so**

- generate a fine k-mesh for the optics part: **x kgen [-so (if case.ksym has been created by symmetso)]**
- change TOT to FERMI in **case.in2c**
- execute **run[sp]_lapw -so -s lapw1 -e lcore** with this fine k-mesh
- run optic: **x opticc -so [-up]**
- run joint: **x joint [-up]**
- run kram: **x kram [-up]**

In cases of non-spinpolarized spin-orbit calculations WITHOUT inversion symmetry one must do some tricks and "mimick" a spinpolarized calculation:

- cp case.vsp case.vspup
- cp case.vsp case.vspdn
- cp case.vectorso case.vectorsoup
- x lapw2 -fermi -so -c
- cp case.weight case.weightup
- cp case.weight case.weightdn
- x optic -so -up
- x joint -up

Due to the "paramagnetic" weight files (which are normalized to 2 electrons per band instead of one) all your results (joint/sigma...) must be divided by a factor of two.

*Note: In spin-polarized cases with spin-orbit only one call to **optic**, **joint** and/or **kram** (either up or down) is necessary, since the spins are not independent any more and both vector-files are used at the same time.*

8.16.2 Dimensioning parameters

The following dimensioning parameters (listed in **param.inc_r** and **param.inc_c**) are used:

LMAX	highest l+1 in basis function inside sphere (reducing LMAX to 4 or 5 may dramatically speed-up optics for large cases, but of course the matrix elements will be truncated and do not have full precision)
LOMAX	highest l for local orbital basis (consistent with input in case.in1)
NRAD	number of radial mesh points
NSYM	order of point group

8.16.3 Input

An example is given below:

```
---------------- top of file: case.inop --------------------
99999 1          : NKMAX, NKFIRST
-5.0 2.0  18     : EMIN, EMAX, NBvalMAX
XMCD  1  L23      : optional line: for  XMCD of 1st atom and L23 spectrum
2                : number of choices (columns in *symmat)
1                : Re xx
3                : Re zz
OFF              : ON/OFF   writes MME to unit 4
------------------ bottom of file -------------------------
```

Interpretive comments on this file are as follows:

line 1: free format

nkmax, nkfirst	maximal number of k-points , number of k-point to start calculation

line 2: free format

emin, emax	absolute energy range (Ry) for which matrix elements should be calculated
nbvalmax	optional input. Setting this to the number of occupied bands (see case.output2) will reduce cpu-time of optics (for large cases, MM only between occupied and empty bands)

line 3: `optional line, must be omitted for ''normal'' optic`; free format

XMCD	fixed keyword to indicate XMCD calculation. You should also use NCOL=6
natom	atom number (from **`case.struct`** file) for which XMCD should be calculated
edge	specify the edge: must be K, L1, L23, M1, M23 or M45

line 3+: free format

ncol	number of choices (columns in case.symmat)

line 4+: free format

icol	column to select. Choices are: 1 ... Re $< x >< x >$ 2 ... Re $< y >< y >$ 3 ... Re $< z >< z >$ 4 ... Re $< x >< y >$ 5 ... Re $< x >< z >$ 6 ... Re $< y >< z >$ 7 ... Im $< x >< y >$ 8 ... Im $< x >< z >$ 9 ... Im $< y >< z >$ Options 7-9 apply only in presence of SO, options 4-6 only in non-orthogonal cases.

line 5: free format
IMME, NATOMS (optional input)

IMME	OFF/ON; optionally prints unsquared momentum matrix elements to unit 4
NATOMS	number of atoms for which the opt. matrix elements should be calculated (The index of the atoms is read in the next line). Please note, that since we need the squared matrix elements, the sum of ϵ_2 using atom "1" and atom "2" separately is NOT the same as using atom "1 and 2" together, since we miss crossterms. Nevertheless this can be a useful option to analyze the origin of certain peaks in ϵ_2. I recommend to repeat this analysis for all possible combinations, and also for a list of "all" atoms, since this shows the effect of the interstitial (and crossterms involving the interstitial).

line 6: (optional) free format

IATOMS	List of NMODUS atoms for which the opt. matrix elements should be calculated (see above).

8.17 JOINT (Joint Density of States)

This program was contributed by:

Claudia Ambrosch-Draxl
Atomistic Modelling and Design of Materials
University Leoben
A-8700 Leoben, AUSTRIA
email: cad@unileoben.ac.at
Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This program carries out the BZ integration using the momentum matrix elements **case.symmat** calculated before by **optic**. The interband or the intraband contributions to the imaginary part of the dielectric tensor (ϵ_2) can be computed. Alternatively, the DOS or the joint DOS can be derived.

The output in **case.joint** can be plotted with any xy-plotting package or **opticplot_lapw** or **Curve_lapw**.

Warning: Negative values for ϵ_2 may occur due to negative weights in Blöchl's tetrahedron method.

For optional XMCD calculations (see OPTICS) an integration of the Brillouin zone is carried out using the momentum matrix elements from **case.symmat1up** and **case.symmat2up** (if both edges are present, otherwise only from **case.symmat1up**). The broadened and unbroadened spectra are written in files **case.xmcd** and **case.rawxmcd**: in these files, the first coloumn is the energy mesh, the second and third coloumns the left and right polarized absorption spectra, the fourth column the XMCD and the last is the XAS. For $L_{2,3}$, $M_{2,3}$, and $M_{4,5}$ edges, the broadened and unbroadened spectra for the single edges (useful for the application of Carra's and Thole's sum rules) are stored in **case.broad1** and **case.broad2** and **case.raw1** and **case.raw2**, respectively, where "1" and "2" are refererred to the higher and lower energy core state.

8.17.1 Execution

The program **joint** is executed by invoking the command:

joint joint.def or **x joint [-up|dn]**

8.17.2 Dimensioning parameters

The following parameter is listend in files **param.inc**:

NSYM	order of point group
MG0	number of columns (usually 9)

8.17.3 Input

An example is given below:

```
---------------- top of file: case.injoint -----------------------
    1  9999  8               : LOWER,UPPER,upper-valence BANDINDEX
```

```
  -0.0000    0.00100   2.0000 : EMIN DE EMAX FOR ENERGYGRID IN ryd
eV                             : output units  eV / ryd
XMCD                             : omitt these 4 lines for non-XMCD
 -49.88 -50.80                   : core energies in Ry (grep :2P case.scfc)
   1.6    0.6                    : core-hole broadening (eV) for both core states
   0.1                           : spectrometer broadening (eV)
    4                          : SWITCH
    2                          : NUMBER OF COLUMNS
   0.1  0.1  0.3               : BROADENING (FOR DRUDE MODEL - switch 6,7)
------------------ bottom of file -------------------------------
```

Interpretive comments on this file are as follows:

line 1: free format

b1, b2, b3	lower, upper and (optional) upper-valence band-index (Setting b3 may allow for additional analysis (restricting the occupied bands from b1-b3) and in big cases it will reduce memory requirements. Otherwise set b3 equal b2)

line 2: free format

emin, de, emax	Energy window and increment in Ry (emin must not be negative)

line 3: free format

units	eV	output in units of eV
	Ry	output in units of Ry

line 4: `optional line for XMCD, must be omitted for ``normal'' optic`; free format

XMCD	keyword for XMCD calculation, requires 3 more lines

line 4xmcd: `must be omitted for ``normal'' optic`; free format

E_core1, E_core2	lower and higher core energies (in Ry, get them using eg. "**`grep :2P case.scf`**")

line 4xmcd: `must be omitted for ``normal'' optic`; free format

broad_core1, broad_core2	lifetime broadening (eV) of lower and higher core state

line 4xmcd: `must be omitted for ``normal'' optic`; free format

broad	spectrometer (Gaussian) broadening (eV)

line 4+: free format

switch	0	joint DOS for each band combination
	1	joint DOS as sum over all band combinations
	2	DOS for each band
	3	DOS as sum over all bands
	4	imaginary part of the dielectric tensor (ϵ_2)

5	imaginary part of the dielectric tensor for each band combination
6	intraband contributions: number of "free" electrons per unit cell assuming bare electron mass (calculated around $E_F \pm 10 * de$ as defined in input line 4), plasma-frequency
7	in addition to switch 6 the contributions from different bands to the plasma frequency are analyzed.

line 5: free format

ncol	number of columns

line 6: free format
broadening

x,y,z	broadening parameters (in units defined in line 3) for Drude-model

The band analysis for all options (switches 0, 2, 5, and 7) has been improved: For each tensor component additional files are created, where each column contains the contributions from a single band or band combination. The file names are e.g. **.Im_eps_xx_1**, **.Im_eps_xx_2**, or **.jdos_1** etc. where the number of files depend on the number of bands/band combinations.

Warning: The number of band combinations might be quite large!

8.18 KRAM (Kramers-Kronig transformation)

This program was contributed by:

Claudia Ambrosch-Draxl
Atomistic Modelling and Design of Materials
University Leoben
A-8700 Leoben, AUSTRIA
email: cad@unileoben.ac.at
Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

The Kramers-Kronig analysis is carried out for the actual number of columns contained in the **case.joint[up|dn]** file. For each real component its imaginary counterpart is created and vice versa. All dielectric tensor components can be found in file **case.epsilon[up|dn]**. The real and imaginary parts of the optical conductivity (in $10^{15}/s$) are written to file **case.sigmak[up|dn]**. In addition, file **case.absorp** contains the real parts of the optical conductivity (in $1/(\Omega cm)$ and the absorption coefficients. The loss function is also calculated (**case.eloss**), where for the previously calculated Plasma-frequency the intraband contributions can be added.

Please note, that for spin-polarized calculations, the Kramers-Kronig analysis is NOT really additive, i.e. most quantities (like ϵ_1) cannot be obtained by simply adding the spin-up and dn results to get the total contribution (see equations in Ambrosch 06). Thus, one should add up both spin contributions of ϵ_2 (in **case.jointup** and **case.jointdn**) using **addjoint-updn_lapw** (this will produce **case.joint**) before calling (non-spinpolarized) **x kram**.

The 3 sumrules are also checked and written to **case.sumrules**.

The output in **case.epsilon[up|dn]** and **case.sigmak[up|dn]** can be plotted with any xy-plotting package, **opticplot_lapw** or the "OPTIC"-task in **w2web**.

8.18.1 Execution

The program **kram** is executed by invoking the command:

kram kram.def or **x kram [-up|dn]**

8.18.2 Dimensioning parameters

The following parameters are listed in files **param.inc**:

MAXDE	maximum number of points in energy mesh
MPOL	fixed at 6

8.18.3 Input

An example is given below:

```
---------------- top of file: case.inkram -----------------------
 0.0    gamma for Lorentz broadening (in units selected in joint)
 0.0    energy shift (scissors operator) (in units selected in joint)
  1      add intraband contributions? yes/no: 1/0
 12.60   plasma frequencies (for each ``column'' in case.injoint)
  0.20   Gammas for Drude terms (for each ``column'' in case.injoint)
------------------- bottom of file --------------------------------
```

Interpretive comments on this file are as follows:

line 1: free format

EGAMM		Lorentz broadening (in energy units selected in joint)

line 2: free format

ESHIFT		Energy shift (scissors operator) (in energy units selected in joint)

line 3: free format

INTRA	0	Intraband contributions are not added
	1	Intraband contributions are added (requires plasma-frequencies calculated by **joint** using switch "6"))

line 4: free format

EPL	Plasma-frequencies (calculated by **joint** using SWITCH=6 for all columns)

line 5: free format

EDRU	Broadening for Drude terms (for all columns)

8.19 DIPAN (Dipolar anisotropies)

This program was contributed by:

P. Novák
Inst. of Physics, Acad.Science, Prague, Czeck Republic
email: novakp@fzu.cz

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This program calculates the magnetic dipolar hyperfine field and the dipolar magnetocrystalline anisotropy by a direct lattice summation over the magnetic moments of all sites.

According to Wikipedia

$$\vec{B} = \frac{\mu_0 \mu}{4\pi r^3}[3(\vec{n}\hat{r})\hat{r} - \vec{n}] \tag{8.1}$$

where $\hat{r} = \vec{r}/r$.
$\vec{n} = \vec{M}/M$ is direction of magnetization.
μ_0 is permeability of free space; $\mu_0 = 4\pi 10^{-7}$ H/m.
$\vec{B}$ is the dipolar field in T.
$\vec{\mu}$ is magnetic dipolar moment in Am^2 = J/T, assumed to be parallel to $\vec{n}$.
r is in m.
We want to express μ in Bohr magnetons μ_B=9.274078.10^{-24} J/T and
r in atomic units for length a_0 (Bohr radius) a_0=5.2917706.10^{-11} m.
Inserting in (1) gives

$$\vec{B} = 6.258463 \frac{\mu(\mu_B)}{r(\text{a.u.})^3}[3(\vec{n}\hat{r})\hat{r} - \vec{n}]. \tag{8.2}$$

Total dipolar field acting on atom i is given by the lattice sum

$$\vec{B_i} = 6.258463 \sum_j \frac{\mu_j}{r_j^3}[3(\vec{n}\hat{r_j})\hat{r_j} - \vec{n}]. \tag{8.3}$$

Dipolar anisotropy energy is given by the sum

$$E_{an} = -\frac{1}{2V}\sum_j \vec{B}_j \vec{\mu}_j \tag{8.4}$$

when the sum is over atoms in the unit cell, V is the unit cell volume, Factor $1/2$ appears because of the double summation.

Expressing B_j in T, μ_j in μ_B and V in (a.u.)^3 gives

$$E_{an}(\text{J/m}^3) = -\frac{3.129232.10^7}{V(\text{a.u.})^3}\sum_j \vec{B}_j(\text{T})\vec{\mu}_j(\mu_\text{B}) \tag{8.5}$$

8.19.1 Execution

The program **dipan** is executed by invoking the command:

dipan dipan.def or **x dipan**

8.19.2 Dimensioning parameters

The following parameters are listed in files **dipan.f**:

NATO	number of inequivalent atoms in unit cell
NDIF	total number of atoms in unit cell

8.19.3 Input

An example is given below:

```
---------------- top of file: case.indipan -----------------------
 160.   0                     Rmax (a.u.),  ipr (printing option)
 -0.26                        Magnetic moment of 1s atom (Y) in mu_B
  1.525                       Magnetic moment of 2nd atom  (Co(2c))
  1.529                       Magnetic moments of 3rd atom (Co(3g)) in mu_B
 1381.                        Volume in a.u.**(-3)
 2                            ndir: numder of magnetization directions
 0. 0. 1.                     first direction for the magnetization
 1. 1. 0.                     second direction
------------------ bottom of file -------------------------------
```

Interpretive comments on this file are as follows:

line 1: free format
Rmax, IPR

Rmax	max distance (bohr) for lattice summation. Vary it for convergence check.
IPR	Print switch. IPR=2 produces very large files **case.outputdipan** and **case.nn_dipan**

line 2: free format

mm	Magnetic moment (μ_B) of first atom

line 2 must be repeated for every non-equivalent atom in the unit cell
line 3: free format

VOLUME	Unit cell volume in bohr**3 (grep :VOL case.scf)

line 4: free format

NDIR	number of magnetization directions for which the dipolar contributions will be calculated. For $NDIR > 1$ the differences $E_{an}(dir_i) - E_{an}(dir_j)$ are also calculated.

line 5: free format

h,k,l	direction of magnetization

line 5 must be repeated NDIR times

8.20 FSGEN (Fermi-surface generation)

Unfortunately there is no really versatile tool for Fermi surface generation or analyzing FS properties. We have collected here a series of small programs together with some description on how to proceed to generate 2D-Fermisurfaces within WIEN.

- As usually, you have to run an scf cycle and determine a good Fermi-energy. "Good" means here a Fermi-energy coming from a calculation with a dense k-mesh.
- You should than create a mesh within a plane of the BZ, where you want to plot the FS. Some utility programs like **sc_fs_mesh**, (fcc, bcc, cxz_mon and hex are also available) may help you here, but only some planes of the BZ have been implemented so far. Please check these simple programs and modify them according to your needs. Copy the generated k-mesh **fort.2** to **case.klist**.
- Run **lapw1** with this k-mesh.
- Run **spaghetti** with input-options such that it prints the bands which intersect EF to **case.spaghetti_ene** (line 10, see sec. 8.3)
- Edit **case.spaghetti_ene** and insert a line at the top:
  NX, NY, x-len, y-len, NXinter, NYinter, Invers, Flip
  where
  NX, NY are the number of points in the two directions
  x-len, y-len are the length of the two directions of the plane (in bohr^{-1}, you can find this in **case.spaghetti_ene**)
  NXinter, NYinter: interpolated mesh, e.g. 2*NX-1
  Invers: 0/1: mirrors x,y
  FLIP: 0/1: flips x,y to y,x
- Run **spagh2rho < case.spaghetti_ene** to convert from this format into a format which is compatible with the **case.rho** file used for charge density plotting. It generates files **fort.11**, **fort.12**, ... (for each band separately) and you should use your favorite plotting program to generate a contourplot of the FS (by using a contourlevel = 0). Alternatively you can use for plotting:
- Run **fsgen_lapw 11 xx save_filename**, which is a small shell script that can plot all fermi surfaces using the data-files **fort.11**, **fort.12**, ... **fort.xx** generated in the previous steps. It requires the public domain package **pgplot** and the contour-plot program **plotgenc**. (The latter can be obtained from **http://www.wien2k.at/reg_user/unsupported/**, but you must have installed the **pgplot** library before.)

9 Utility Programs

Contents

9.1 symmetso

This program helps to setup spin-orbit calculations in magnetic systems. Since SO may break symmetry in certain spacegroups, it classifies your symmetry operations into operations **A**, which do not invert the magnetization (identity, inversion, rotations with the rotation axis parallel to magnetization), **B**, which invert it (mirror planes) and **C**, which change the magnetization in some other way. (Note: magnetization is a result of a circular current, or equivalently, an axial vector resulting from a vector product $\hat{z} \sim \hat{x} \times \hat{y}$). **symmetso** will keep all A-type and throw away all C-type symmetry operations. Depending on the presence of inversion symmetry it will keep (inversion is present) or remove the B-type operations. Finally, **symmetso** uses the remaining

symmetry operations to check/generate equivalent atomic positions (it can happen that some equivalent atoms become non-equivalent after inclusion of SO interaction).

In essence, it reads your **case.struct** and **case.inso** (for the direction of magnetization) files and creates an ordered **case.struct_orb** file with proper symmetry and equivalent atoms. It also generates a file **case.ksym**, which is a struct file with valid operations to generate a proper k-mesh using **''x kgen -so''**. In addition proper input files **case.in1, case.in2, case.inc, case.vspup/dn, case.vnsup/dn, case.clmsum, case.clmup/dn** are generated, so that you can continue with **runsp -so** without any further changes.

9.1.1 Execution

The program **symmetso** is executed by invoking the command:

symmetso symmetso.def or **x symmetso [-c]**

Usually it is called from the script **initso_lapw** and thus needs not to be invoked manually.

9.2 pairhess

This program was contributed by:

James Rondinelli, Bin Deng and Laurence Marks
Dept. Materials Science and Engineering
Northwestern University
Evanston, USA
l-marks@northwestern.edu

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This program creates an approximate hessian matrix (in **.minpair**) for structure minimization using the PORT option. It uses a harmonic model with exponentially decaying bond strenght and in many cases reduces the number of geometry steps during **min_lapw** significantly. It is described in detail in Rondinelli et al. 2006.

For its usage see the comments in sect. 5.3.2.

9.2.1 Execution

The program **pairhess** is executed by invoking the command:

pairhess pairhess.def or **x pairhess [-copy]**

The switch **-copy** copies **.minpair** to **.minrestart** and **.min_hess**, which are needed in **min_lapw**.

9.2.2 Dimensioning parameters

The following parameters are used in **param.inc**:

NATMAX	max. number of atoms)
NEIGMAX	max number of neighbours

9.2.3 Input

pairhess uses an optional input file **case.inpair**, which is needed only for an experienced user for better tailoring of certain default parameters.

An example is given below:

```
---------------- top of file: case.inpair -----------------------
10.0 2.0 0.25  (Rmax, Decay, ReScale)
0.05 1.0 0     (Cutoff, Diag, mode)
0.2            (ZWEIGHT
```

Interpretive comments on this file are as follows:

line 1: free format

RMAX	Maximum distance (a.u.) for considering neighbors. 8-12 is good.
DECAY	Exponential decay applied to neighbors when calculating the pairwise bond strenghts. 1.5-2.5 is reasonable.
RESCALE	A scaling term to multiply the pairwise hessian by. This number is rather important; 0.25 appears to be best for a system with soft modes, 0.35 for a stiffer system. You can save substantial time by adjusting RESCALE so it is approximately correct using a .min_hess from a previous run (adjust until numbers for similar multiplicities are similar), or by adjusting the frequencies (see also eigenhess).

line 2: free format

CUTOFF	When the weighting (via an exponential decay) becomes smaller than this number the pairwise bonds are ignored.
DIAG	The value to multiply a unitary matrix by, this is added to the hessian estimate
MODE	0: Spring model; [1: harmonic model; not so good]

line 3: free format

ZWEIGHT	Atomic number weight for bonds of form exp(-Z*ZWeight). Values of 0.1-0.2 are reasonable. The default is 0.1; a negative number (e.g. -1) turns this off.

9.3 eigenhess

This program was contributed by:

Laurence Marks
Dept. Materials Science and Engineering
Northwestern University
Evanston, USA
l-marks@northwestern.edu
Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This program analyses / manipulates **.min_hess**, which was created by a structural minimization using **min_lapw** and the "PORT" option. In particular, such an analysis can yield approximate vibrational frequencies and corresponding eigenmodes, which eventually can give a hint about a dynamically unstable structure (imaginary frequencies). Some more description is given in $WIENROOT/SRC_pairhess/README.

The program **eigenhess** is executed by invoking the command:

x eigenhess

9.4 patchsymm

This program was contributed by:

James Rondinelli, Bin Deng and Laurence Marks
Dept. Materials Science and Engineering
Northwestern University
Evanston, USA
l-marks@northwestern.edu
Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This program performs a symmetry check on the positions and produces a new struct file **case.struct_new**. It is useful in case something went wrong during **min_lapw** (rounding errors of positions) or the cif/amc file did not have enough digits (eg. "1/3" was prepresented by "0.33333" only). The file case.outputpatch gives information on how parameters changed.

9.4.1 Execution

The program **patchsymm** is executed by invoking the command:

patchsymm patchsymm.def or **x patchsymm**

9.5 afminput

This program creates the inputfile **case.inclmcopy_st** for the program **clmcopy**, which copies spin-up densities of atom i to spin-down densities of the related antiferromagnetic atom j and vice versa in an anti-ferromagnetic system. It uses a symmetry operation to find out how and which atomic densities must be interchanged and how the Fourier coefficients of the density transform. It is based on the ideas of Manuel Perez-Mato (Bilbao, Spain).

See $WIENROOT/SRC_afminput/afminput_test for several examples.

The best way is to supply a file **case.struct_supergroup**, which is the struct file of the nonmagnetic supergroup. If the two spacegroups are "TRANSLATIONENGLEICH", it will find out automatically the proper symmetry operation. *Please note, this automatic way works only when the coordinate system remains identical. In some cases sgroup may interchange eg. the y and z axis. In such cases reverse this change, both, for the lattice parameters as well as for all positions, set NSYM=0 and run init_lapw again (ignoring any suggestion of sgroup).*

If the two spacegroups are "KLASSENGLEICH" (i.e. have the same number of symmetry operations), you will be asked to supply a translation which transforms the AF atoms into each other. A typical example would be bcc Cr: the bcc supergroup and the AF subgroup (simple cubic) have both 48 symmetry operations and the proper translation is (0.5,0.5,0.5).

Finally, if you don't give **case.struct_supergroup**, you have to supply a symmetry operation (rotation + non-primitive translation) as input. For bcc Cr or the famous NiO-AFII structure this would be simply

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

Please see the comments in sect. 4.5.4 on how to proceed in detail for AFM calculations and find further examples in **SRC_afminput**.

9.5.1 Execution

The program **afminput** is executed by invoking the command:

afminput afminput.def or **x afminput**

9.5.2 Dimensioning parameters

The following parameters are used:

NCOM	number of LM components in the density (in **param.inc**)
LMAX	max l for LM expansion of the density (in **param.inc**).

9.6 clmcopy

This program generates the spin-dn density (**case.clmdn**) from a given spin-up density (**case.clmup**) according to rules and symmetry operations in **case.inclmcopy** (generated earlier by **afminput**) for an AFM calculation.

Please see the comments in sect. 4.5.4 on how to proceed in detail for AFM calculations.

9.6.1 Execution

The program **clmcopy** is executed by invoking the command:

clmcopy clmcopy.def or **x clmcopy**

9.6.2 Dimensioning parameters

The following parameters are used in **param.inc**:

NCOM	number of LM components in the density
NRAD	number of radial mesh points
NSYM	number of symmetryoperations

9.6.3 Input

An example is given below:

```
---------------- top of file: case.inclmcopy -----------------------
   2                                              NUMBER of ATOMS to CHANGE
   1   2                                          INTERCHANGE these ATOMS
 -1.00000000000  0.00000000000  0.00000000000     SYMMETRY OPERATION
  0.00000000000 -1.00000000000  0.00000000000
  0.00000000000  0.00000000000 -1.00000000000
   0                                              NUMBER of LM to CHANGE SIGN
   3   4                                          INTERCHANGE these ATOMS
 -1.00000000000  0.00000000000  0.00000000000     SYMMETRY OPERATION
  0.00000000000 -1.00000000000  0.00000000000
  0.00000000000  0.00000000000 -1.00000000000
   9                                              NUMBER of LM to CHANGE SIGN
  1 0    1 0 -1.00
  3 0    3 0 -1.00
  3 2    3 2 -1.00
 -3 2   -3 2 -1.00
  5 0    5 0 -1.00
  5 2    5 2 -1.00
 -5 2   -5 2 -1.00
  5 4    5 4 -1.00
 -5 4   -5 4 -1.00
   1   0   0   0.50000
   0   1   0   0.00000
   0   0   1   0.50000
```

Interpretive comments on this file are as follows:

line 1: free format

NATOM	Number of atoms for which rules for copying the density will be defined

line 2: free format

N1, N2	Interchange spin-up and dn densities of atoms N1 and N2

line 3-5: free format

SYM	Symmetry operation for atom N1 to rotate into N2 (without translational part)

line 6: free format

NLM	Number of LM values, for which you have to change the sign when swapping up and dn-densities

line 7ff: free format

L1,M1,L2,M2,Fac	NLM pairs of L1,M1 (spin-up), which change into L2,M2 (spin-dn) and the respecting CLMs are multiplied by Fac

Lines 2-7ff have to be repeated NATOM times.

line 8-10: free format

SYM0	Symmetry operation (one of the operations of the NM-supergroup missing in the AFM-subgroup (transfers spin-up into spin-dn atom)

9.7 reformat

To produce a surface plot of the electron density using **rhoplot_lapw** (which is an interface to **gnuplot**), data from the file **case.rho** created by **lapw5** must be converted using **reformat**

The sources of the program **reformat.c** are supplied in **SRC_reformat**.

9.8 hex2rhomb and rhomb_in5

hex2rhomb interactively converts the positions of an atom from hexagonal to rhombohedral coordinates (needed in **case.struct**).

rhomb_in5 interactively helps to generate input **case.in5** for density plots with **lapw5** for rhombohedral systems. It defines a plane as needed in the input file when you specify 3 atoms of that plane.

The sources of these programs are supplied in **SRC_trig**.

9.9 plane

plane helps to generate **case.in5** for density plots with **lapw5** (for orthogonal and hex lattices only). The plane will be specified by 3 atoms and you need an auxiliary file **plane.input**, which contains:

```
a,b,c     # lattice parameters
x0,y0,z0  # position of atom (fractional coordinates), which will be centered in the plot
x1,y1,z1  # position of atom, which will be ``below'' the centered atom
x2,y2,z2  # position of atom, which will show to the ``left''
x1,y1     # lenght (in bohr) of plot in x and y direction.
'P'       # defines lattice, either P (carthesian coordinates) or H (hexagonal) supported
```

The source of this program is supplied in **SRC_trig**.

9.10 add_columns

add_columns reads a sequence of pairs of 2 numbers (form stdin), adds them together and prints the sum to stdout. If you have two columns of numbers in 2 files (eg. in colup and coldn) you can add them using:

```
paste colup coldn | add_columns > col
```

The source of this program is supplied in **SRC_trig**.

9.11 clminter

clminter interpolates the density in **case.clmsum/up/dn** to a new radial mesh as defined in **case.struct_new**. This utility is usefull when you run a structural minimization (**min_lapw**), some atoms start to overlap and you have to reduce RMT (the size of the atomic spheres) of certain atoms. In such a case:

- save the calculations
- generate **case.struct_new** with modified RMTs
- **x clminter**
- in spinpolarized case repeat this line with -up and -dn switches
- cp case.struct_new case.struct
- cp case.clmsum_new case.clmsum
- eventually copy also case.clmup/dn files)
- **run_lapw**; (it will probably take some iterations until you reach scf again, but it should be much faster than starting with **init_lapw**)

Note: Please be aware the the total energy will change with modified RMT (by some constant) and you must not compare energies comming from different RMTs (but most likely you can determine the constant shift by repeating (at least) ONE calculation with identical structure but different RMTs).

The source of this program is supplied in **SRC_trig**.

9.12 eosfit

Small program to calculate the Equation of States (EOS; Equilibrium volume V_0, Bulk modulus B_0 and it's derivative B_0'. The Murnaghan (1944), the Birch-Murnaghan and the EOS2 equation of states are supported. It relies on the file **case.vol** (containing lines with "volume, E-tot", usually created from **w2web** using "Volume optimization"), or alternatively is called from **eplot_lapw** using **case.analysis** (see 5.7.1 and 5.3.1).

The sources are supplied in **SRC_eosfit**.

9.13 eosfit6

Nonlinear least squares fit (using PORT routines) for a parabolic fit of the energy vs. 2-4 dim. lattice parameters. It requires **case.ene** and **case.latparam**, usually generated by **parabolfit_lapw**. It can optionally produce **case.enefit**, which contains energies on a

specified grid for plotting purposes (in 2D same format as **case.rho**, which can be used in contourplot programs). (See 5.3.1).

The sources are supplied in **SRC_eosfit6**.

9.14 spacegroup

This program was contributed by:

Vaclav Petricek
Institute of Physics
Academy of Sciences of the Czech Republic
Na Slovance 2
182 21 Praha (Prague) 8
Czech Republic
petricek@fzu.cz

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

Interactive program to generate equivalent positions for a given spacegroup and lattice. The program is also used internally from **w2web** to generate positions when selecting spacegroups in the **StructGen**.

9.15 join_vectorfiles

This program was contributed by:

Phillipp Wissgott
Institute of Solid State Physics
TU Vienna
wissgott@ifp.tuwien.ac.at

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

Interactive program to combine parallel vector and energy files (**case.vector_xx** and **case.energy_xx**) into single files (**case.vector** and **case.energy**).

Executed by:

- x join_vectorfiles [-up/-dn/-so/-soup/-sodn] [-c] case number_of_parallel_files_to_join

9.16 arrows

This program was contributed by:

Evgeniya Kabliman
Institute for MaterialsChemistry
TU Vienna
evgeniya@theochem.tuwien.ac.at
Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

Small program which together with **Xcrysden** allows to display the "forces acting on all atoms" or the "differences between two structures" using arrows which indicate the movement of the atoms. The recommended sequence to visualize **forces** is:

- Prepare (copy) a struct and scf file with the initial structure using the names **case_initial.struct** and **case_initial.scf**.
- View **case_initial.struct** in **Xcrysden** and "File/Save as xsf-structure" with the name **case_initial.xsf**.
- **x arrows**
- View the resulting **case_forces.xsf** using: **xcrysden --xsf case_forces.xsf**. Switch on "Display/Forces" and adjust the length of the arrows in "Modify/Force-settings".

while differences between the inital and relaxed structure can be viewed by:

- Prepare (copy) two struct files with the initial and the relaxed structure using the names: **case_initial.struct** and **case_final.struct**.
- View **case_initial.struct** in **Xcrysden** and "File/Save as xsf-structure" with the name **case_initial.xsf**.
- **x -delta arrows**
- View the resulting **case_delta.xsf** using: **xcrysden --xsf case_delta.xsf**. Switch on "Display/Forces" and adjust the length of the arrows in "Modify/Force-settings".

9.17 xyz2struct

xyz2struct reads "atomlabel,x,y,z"-data from **case.xyz** and writes them into **xyz2struct.struct**. You may have to edit the xyz-file and insert a few lines at the top:

ANG(default)/BOHR; F/C (fractional or carth. coordinates) and L (lattice information, see example)

a,b,c lattice parameters, or when L was specified a scaling constant and the bravais matrix.

Since xyz data contain no symmetry information, all atoms with the same "label" will be treated as equivalent. The nuclear charges ZZ will not be given and you have to insert them manually or use **w2web-StructGen**.

It is executed using:

```
xyz2struct < case.xyz
```

(I recommend this program only for cases with many non-equivalent atoms and (almost) no symmetry. If you have spacegroup-information it is probably easier to use **StructGen** and copy/paste of the positions).

A proper **case.xyz** file looks like:

```
ang
7.47700   7.47700   7.47700
B      4.98466667     1.24616667     0.00000000
C      6.23083333     2.49233333     0.00000000
.....
```

or

```
BOHR F L
17.47700
   0.470724637   0.492808141   0.000000000
  -0.471118220   0.493012774   0.000000000
   0.000000000   0.000000000   0.680559876
B      0.00000000     0.00000000     0.00000000
C      0.14300000     0.14300000     0.25000000
.....
```

9.18 cif2struct

cif2struct reads structural data in cif-format from **case.cif** and writes them into **case.struct**. It is executed using:

cif2struct case.cif

The required cif files can be for example be obtained from Cystallographic databases (e.g. the Inorganic Crystal Structure DataBase ICSD) or from other programs.

Alternatively, **cif2struct** can work with **case.txt**, which contains the following data:

```
a                                       # a..Ang, b..Bohr
  0.0  0.0  0.0                         # shift of origin
  4.7554 4.7554 12.991 90. 90. 120.     # a,b,c,angles
'R-3c'                                  # spacegroup-symbol (see \STRUCTGEN{})
'Al'                                    # atom-name
 0.0000000 0.0000000 0.3520000          # atomic position
'O'                                     # ...
 0.3063000 0.0000000 0.2500000          # ...
...
```

9.19 struct2cif

struct2cif creates a cif-file **struct.cif** from **case.struct**. It is executed using:

x struct2cif

and will ask for the name of a struct file and a spacegroup. It was contributed by F. Boucher (Florent.Boucher@cnrs-imn.fr) and L.D.Marks (L-marks@northwestern.edu). There is also a similar program **struct2xyz** available.

9.20 StructGen of w2web

The new **StructGen** helps to generate the master input file **case.struct**. It has the following additional features:

- automatic conversion from/to Å and Bohr
- Use spacegroup information (in conjunction with the **spacegroup** program (see 9.14 to generate equivalent positions)

- built in calculator to carry out simple arithmetic operations to specify the position pameters (of the equivalent atoms). Each position of equivalent atoms can be entered as a number, a fraction (e.g. $1/3$) or a simple expression (e.g. $0.21 + 1/3$). The first position defines the variables x, y and z, which can be using in expression defining the other positions (e.g. $-y$, x, $-z + 1/2$).

9.21 supercell

This program helps to generate supercells from a regular **WIEN2k**-struct file.

It asks interactively for the name of the original struct file and the number of cells in x, y, and z direction. (Only integers are allowed, thus no rotations by 45^o like sqrt(2) x sqrt(2) cells are supported yet).

If symmetry permits, one can change the target lattice to P, B or F centered lattices, which allows to increase the number of atoms in these supercells by a factor of 2, 4, 8, ...

Rhombohedral (R) lattices are converted automatically into H (hexagonal) lattices, which are 3 times larger than the original cell.

If the target lattice is P, one can add some vacuum in each direction for surface slabs (or chains or isolated molecules) and also add a "top"-layer (repeat the atoms with z=0 at z=1).

You can define an optional shift in x,y,z direction for all the atoms in the cell. (This might be usefull if you want to arrange the atoms in a certain way, eg. you may want to create a surface slab such that it is centered around z=0.5 (and not z=0), so that plotting programs (xcrysden) produce nicer pictures of the structure.

For the experienced user a much more flexible (but also more complicated) tool is available, namely the **structeditor** package (see Sect.9.22).

Please note: You cannot make calculations with these supercells (except for surfaces) unless you modify the created supercell-struct file. You must break the symmetry by introducing some distortions (e.g. for a frozen phonon) or replace one atom by an impurity/vacancy, ...

9.21.1 Execution

The program **supercell** is executed by invoking the command:

supercell or **x supercell**

9.22 structeditor

This program was contributed by:

Robert Laskowski
email: rolask@theochem.tuwien.ac.at

Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This package helps to manipulate structures. Usually one would start from an appropriate (simple) **case.struct** file, and this tool allows to add or manipulate atoms (with or without symmetry considerations), or generate arbitrary supercells or surfaces. It is commandline driven and targeted for the more experienced user, who "knows what he wants to do" and is just looking for a convenient tool.

It consists of a couple of octave (mathlab) routines and some fortran code, thus it requires **octave** (the free **mathlab** version) and for visualization the **opendx** package (http://www.octave.org and http://www.opendx.org).

A full documentation and some examples can be found in **$WIENROOT/SRC_structeditor/doc**, but the main commands are:

```
a2adist            * calculates  distance between atoms
mina2adist         * calculates minimum distance between atoms
addatom            * adds an atom to the structure
addeqatom          * adds an atom and all equivalent
copyatom           * creates a copy of an atom
getaname           * converts atomic number into atomic symbol
getar0             * calculates r0 from atomic number
getazz             * converts atomic name into atomic number
loadstruct         * reads Wien2k structfile
movealla           * moves all atoms with vector vec
replaceatom        * replaces an atom with other atom
replaceeqatoms     * replaces an atom and all equivalent with other atoms
rmatom             * removes an atom
rmeqatoms          * removes an atom and all equivalent
savestruct         * saves crystal structure
showequivalent     * outputs list of equivalent atoms
showstruct         * displays structure (using DX)
smultatom          * creates symmetry equivalent positions
sshift             * symmetric shifts of equivalent atoms
makeconventional   * converts structure into the conventional form
makeprimitive      * converts structure to the primitive form
makesupercell      * creates supercell
makesurface        * creates surface for a given unitcell
```

9.22.1 Execution

The **structeditor** is invoked within the octave environment:

octave

s=loadstruct("GaN.struct")

# make an orthorhombic supercell and visualize it
a=[1 0 0; 1 1 0; 0 0 2]
sout=makesupercell (s,a);
showstruct(sout);

save it as test.struct
savestruct (sout,"test.struct");

get help on all commands
helpstruct

9.23 Visualization

9.23.1 BALSAC

balsac (Build and Analyze Lattices, Surfaces and Clusters) was written by Klaus Hermann (Fritz-Haber Institut, Berlin). It provides high quality postscript files. In SRC_balsac-utils we provide the following interface programs to convert from **WIEN2k** to **balsac**:

- **str2lat** to convert **case.struct** to **case.lat** (the BALSAC "lat" file).
- **str2plt** to convert **case.struct** to **case.plt** (the BALSAC "plt" file for one unit cell).
- **outnn2plt** to convert **case.outputnn** to **case.plt** (the BALSAC "plt" file for one unit cell). You have to select one atom (central atom) and than all nn-atoms are converted into the plt file.
- In addition converters to the xyz-format (**str2xyz, outnn2xyz**) for other plotting programs are also available.

For an example see figure 3.1 For scientific questions concerning BALSAC please contact Klaus Hermann at hermann@FHI-Berlin.MPG.DE

Balsac is available from:

Garching Innovation GmbH, Mrs. M. Pasecky Hofgartenstr. 8, D-80539 Munich, Germany
Tel.: +49 89 2909190, Fax.: +49 89 29091999
e-mail: gi@ipp.mpg.de
web: http://www.fhi-berlin.mpg.de/th/personal/hermann/balpam.html

9.23.2 XCrysDen

XCrysDen (Kokalj 1999) is a render and analysis package. It has the following features (see also http://www.xcrysden.org/doc/wien.html):

- render and analyze (distances, angles) the crystal structure
- generate k-mesh for bandstructure plots
- generate input and render 2D charge densities
- generate input and render 3D charge densities
- generate input and render Fermi surfaces
- render changes between two structures (original and relaxed) with the help of the **arrows** program (see 9.16)

XCrysDen is available from:

Tone Kokalj
Jozef Stefan Institute, Dept. of Physical and Organic Chemistry
Jamova 39, SI-1000 Ljubljana, Slovenia
Tel.: +386 61 177 3520, Fax: +386 61 177 3811
Tone.Kokalj@ijs.si
http://www.xcrysden.org/

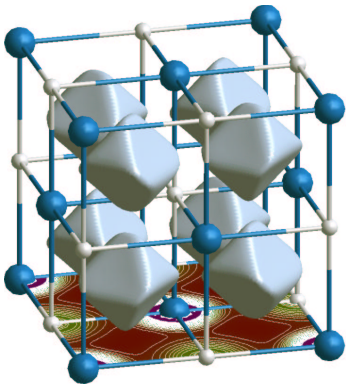

Figure 9.1: 3D electron density in TiC generated with XCrysDen

9.24 Unsupported software

On our website **http://www.wien2k.at/reg_users** you can find a link to **Unsupported software goodies**, where references to various software packages are given. Most of those packages are contributions from WIEN2k-users and you may check this site from time to time if you find some useful tools for you.

In case you develop some goodies yourself and want to share this development with the WIEN2k community, please send an email to **pblaha@theochem.tuwien.ac.at** and we will add it to this page.

10 How to run WIEN2k for selected samples

Three test cases are provided in the **WIEN2k** package. They contain the two starting files **case.struct** and **case.inst** and all the output so that you can compare your results with them.

The test cases are the following (where the names correspond to what was called CASE in the rest of this User's Guide)

TiC
Fccni
TiO2

We recommend to run these test cases (in a different directory) and compare the output to the provided one. All test cases are setup such that the CPU-time remains small (seconds). For real production runs the value of RKMAX in **case.in1** must be increased and a better (denser) k-mesh should be used.

In addition we provide a subdirectory **example_struct_files** were various more complicated struct files can be found.

10.1 TiC

The TiC example is described in detail in chapter 3 (**Quickstart**).

10.2 Fcc Nickel (spin polarized)

Ferromagnetic Nickel is a test case for a spin-polarized calculation. Ni has the atomic configuration $1s^2$, $2s^2$, $2p^6$, $3s^2$, $3p^6$, $3d^8$, $4s^2$ or [Ar] $3d^8$, $4s^2$. We treat the $1s$, $2s$, $2p$ and $3s$ as core states, and $3p$ (as local orbital), $3d$, $4s$ and $4p$ are handled as valence states. In a spin-polarized calculation the file structure and the sequence of programs is different from the non-spin-polarized case (see 4.5.2).

Create a new session and its corresponding directory. Generate the structure with the following data (we can use a large sphere as you will see from the output of **nn**):

Title	fcc Ni
Lattice	F
a	6.7 bohr
b	6.7 bohr
c	6.7 bohr
α, β, γ	90
Atom	Ni, enter position (0,0,0) and RMT = 2.3

Initialize the calculation using the default RKmax and use 3000 k-points (a ferromagnetic metal needs many k-points to yield reasonably converged magnetic moments). Allow for spin-polarization.

Start the scf cycle (**runsp_lapw**) with "-cc 0.0001" (in particular for magnetic systems charge convergence is often the best choice). At the bottom of the converged scf-file (**Fccni.scf**) you find the magnetic moments in the interstital region, inside the sphere and the total moment per cell (only the latter is an "observable", the others depend on the sphere size).

```
:MMINT: MAGNETIC MOMENT IN INTERSTITIAL =   -0.03130
:MMI001: MAGNETIC MOMENT IN SPHERE  1    =    0.66198
:MMTOT: TOTAL MAGNETIC MOMENT IN CELL   =    0.63068
```

10.3 Rutile (TiO_2)

This example shows you how to "optimize internal parameters" and do a k-point parallel calculation.

Create a new session and its corresponding directory. Generate the structure with the following data (we use a smaller O sphere because Ti-d states are harder to converge then O-p):

Title	TiO2
Spacegroup	$P4_2/mnm$ (136)
a	8.682 bohr
b	8.682 bohr
c	5.592 bohr
α, β, γ	90
Atom	Ti, enter position (0,0,0) and RMT = 2.0
Atom	O, enter position (0.3,0.3,0) and RMT = 1.6

StructGenshould automatically add the equivalent positions.

Initialize the calculation using RKmax=6.5 in **tio2.in1_st** and use 100 k-points and a "shift" in **kgen**.

If you have more cpus available (a parallel machine or simply a couple of PCs with a common NFS filesystem, for details see 5.5), you can use "*Execution* ⇩ Run scf", activate the "parallel" button" and "start scf" in **w2web**. This will create and open a **.machines** file and you should insert lines with the proper names of your PCs (possibly use 9 (or 3) processors since we have 9 k-points,). Save this file and click on "*Execution* ⇩ Run scf", activate "-fc 1.0" for force-convergence and "start scf" to submit the scf-cycle.

Alternatively at the command-line you can use the UNIX command

```
cp  $WIENROOT/SRC_templates/.machines .
```

and edit this file. You would start the scf-cycle (in background) simply by typing

```
run_lapw -p -fc 1.0 &
```

During the scf-cycle monitor **tio2.dayfile** and check convergence (:ENE, :DIS, :FGL002), either using "Utils/Analysis" in **w2web**, or **``grep :ENE tio2.scf''**. You should see some convergence of :FGL002 and then a big jump in the final cycle, when the valence-force corrections are added. Only the last force (including this correction) is valid.

Since this force is quite large, you can now optimize the position of the O-atom:

Start the structure minimization in **w2web** using "*Execution* ⇩ mini.positions". This will generate **TiO2.inM**, and you can try option PORT with tolf=1.0 (instead of 2.0), otherwise stay with the default parameters. Repeat "*Execution* ⇩ mini.positions" and start the minimization.

Alternatively you can use

```
min_lapw -p
```

which is identical to:

```
min_lapw -j ``run_lapw -I -fc 1 -p''
```

This will create **TiO2.inM** automatically, call the program **min**, which generates a new struct file using the calculated forces, and continues with the next scf cycle. It will continue until the forces are below 1 mRy/bohr (TiO2.inM) and the final results are not "saved" automatically but can be found in the "current" calculation.

You should watch the minimization (:ENE, :FGL002, :POS002) using the file **TiO2.scf_mini**, which contains the final iteration of each geometry step (see also Sec.5.3.2). If the forces in this file oscillate from plus to minus and seem to diverge, or if they change very little, you can edit **TiO2.inM** (change the method, reduce or increase the stepsize), and remove **TiO2.tmpM** (contains the "history" of the minimization and is used to calculate the velocities of the moving atoms). (This should not be neceaasry for the rutile example, but may occur in more complex minimizations. See comments in Sec. 5.3.2).

The final structural parameter of the O-atom should be close to x=0.304, which compares well with the experimental x=0.305.

10.4 Supercell calculations on TiC

This example shows you how to create a supercell of TiC, which could be used to simulate a TiC-surface or vacancies, impurities or core-holes for X-ray absorption / ELNES spectroscopy. I'll describe the procedure using Unix and WIEN2k commands in an xterm, but of course you can do the same in **w2web**.

Create a new directory, copy the original TiC struct file into it and run **supercell** program:

```
mkdir super
cd super
cp ../TiC/TiC.struct .
x supercell
```

Specify "TiC.struct", a "2x2x2" supercell, "F" lattice (this will create a cell with 16 atoms, you can also create 32 or 64 atom cells using B or P lattice type. Note: surfaces require a P supercell).

```
cp TiC_super.struct super.struct
```

and edit this file to make some changes. You could eg.

- delete an atom (to simulate a vacancy)
- replace an atom by another element (impurity)
- "label" an atom (put a 1 in the 3rd column next to the element name) to make this atom unique (needed eg. for core-holes)
- displace an atom (for phase transitions or phonons)

Note: it is important to make at least one of these chages. Otherwise the initialization will restore the original unit cell (or the calculations will fail later on because symmetry is most likely not correct)

Run **`init_lapw`**. You will see that **`nn`** complains and finds a new set of equivalent atoms (originally all atoms were non-equivalent, but nn finds that some atoms have identical neighbors, thus should be in an equivalent set). Accept the automatically generated struct file and continue. Remember, supercells normally require less k-points than the original small cell.

After the complete initialization you may in principle restore the original struct file (eg. without a displacement) in case you want to "repeat" the undistorted structure in supercell geometry.

For a "core-hole" calculation you would now edit **`super.inc`** and remove one core electron from the desired atom and state (1s or 2p, ...). In addition you should add the missing electron either in **`super.inm`** (background charge) or **`super.in2`** (add it to the valence electrons). In the latter case, you should remove this extra electron AFTER scf and BEFORE calculation of the spectra.

Once this has been done, you could start a scf-cycle (for impurities, vacancies,.. you should most likely also optimize the internal positions).

10.5 Further examples

Further examples can be found on our web-site:

http://www.wien2k.at/events/ws2008/talks/Exercises_08.pdf

Part III

Installation of the WIEN2k package and Dimensioning of programs

11 Installation and Dimensioning

Contents

11.1 Requirements

WIEN2k is written in FORTRAN 90 and requires a UNIX operating system since the programs are linked together via C-shell scripts. It has been implemented successfully on the following computer systems: Intel and AMD based PCs running under Linux, IBM RS6000, HP, SGI, DEC Alpha and SUN. Hardware requirements will change from case to case (small cases with 10 atoms per unit cell can be run on a Pentium-II PC with 128 MB under Linux), but we recommend a more powerful PC or workstation with at least 256 MB (better 512 MM or more) memory and plenty of disk space (a few Gb).

For coarse grain parallization on the k-point level, a cluster of PCs with a 100 Mb/s (better 1Bg/s) network is sufficient. Faster communication (Infiniband) is recommended for the fine grain (single k-point) parallel version.

For Intel (AMD) based systems we recommend the **`Intel ifort compiler`** and the **`Intel mkl library`** (which includes blas, lapack and Scalapack) (see http://www.intel.com). If you have installed **`ifort`** yourself on your local PC, don't forget to configure your environment properly. Add some thing like:

```
source /opt/intel/11.0/074/bin/ifortvars.csh intel64
source /opt/intel/11.0/074/mkl/tools/environment/mklvarsem64t.csh
```

to your **`.cshrc`** file (or similar statements for .bashrc).

In order to use all options and features (such as the new graphical user interface **w2web** or some of its plotting tools) the following public domain program packages in addition to a F90 compiler must be installed:

- perl 5 or higher (for **w2web** only)
- emacs or another editor of your choice
- ghostscript (with jpg support)
- gnuplot (with png support)
- www-browser

- pdf-reader, acroread
- Tcl/Tk-Toolkit (for Xcrysden only)
- MPI+SCALAPACK (for fine grain parallelization only)
- FFTW-2.1.5 (mpi-version for fine grain parallelization only)

Usually these packages should be available on modern systems. If one of these packages is not available it can either be installed from public domain sources (ask your computing center, use the WWW to search for the nearest location of these packages) or the corresponding configuration may be changed (e.g. using vi instead of emacs). Brief installation instructions for mpich and fftw are given below. None of the principal components of **WIEN2k** requires these packages, only **w2web** needs them.

11.1.1 Installation tips for mpich and fftw-2.1.5

This is only a brief guidance, you may need some Linux experience for this.

- Download the mpich1.2.7p1 and fftw-2.1.5 sources from http://www-unix.mcs.anl.gov/mpi/mpich1/ and http://www.fftw.org/download.html (Please note, the fftw-3.x versions are incompatible with fftw-2.x)
- unzip and untar the downloaded file
- Change into the expanded directories and configure the compilation. Define your fortran compiler (setenv FC ifort, or export FC=ifort) and use "./configure −−prefix=/pathname" to configure compilation. /pathname is the directory where the libraries should be installed (could be /opt/local or /usr/local or similar, you will have to specify this path again in the LDFLAGS). For fftw configuration add the "−−enable-mpi" switch.
- make
- make install (if you specified a "system-directory" like /usr/local you must have proper permissions for this step, eg. become root user)
- add the mpi-directory to your path (set path = (/opt/mpich/mpich-1.2.7p1/bin $path))

Optionally, one can also use in the sequential (non-mpi) version of **lapw0** and **lapw2** the fftw routines. However, there is some speedup only when you use the **MKL**-fft routines, not the self-compiled fftw-binaries. The mkl-interface to fftw is not active by default, but you may have to compile it yourself. To do so (syntax ifort12):

- cd $MKLROOT/interfaces/fftw2xf
- make libintel64 (or make libia32)
- in case you do not have icc installed, but use GNU-C (gcc) you must:
  - edit makefile, and remove -D_GNU from the line " CC=gcc -D_GNU" (to remove additional _ from the object names)
  - make libintel64 compiler=gnu
- add -DFFTW2 to FOPT in the **Makefile** of **lapw0** and **lapw2**
- add -lfftw2xf or -lfftw2xf_gnu to R_LIBS in the **Makefile** of **lapw0** and **lapw2**

This may speedup the fft-parts of these programs a bit.

11.2 Installation of WIEN2k

11.2.1 Expanding the WIEN2k distribution

The **WIEN2k** package comes as a single tar file (or you can download about 50 individual tar files separately), which should be placed in a subdirectory which will be your **$WIENROOT** directory

(e.g. **./WIEN2k**). In addition you can download three examples, namely **TiC.tar.gz, TiO2.tar.gz** and **Fccni.tar.gz**.

Uncompress and expand all files using:

tar -xvf wien2k_00.tar (skip this if you downloaded files separately)
gunzip *.gz
chmod +x ./expand_lapw
./expand_lapw

You should have gotten the following directories:

```
./SRC
  SRC_2Doptimize
  SRC_afminput
  SRC_aim
  SRC_arrows
  SRC_balsac-utils
  SRC_broadening
  SRC_cif2struct
  SRC_clmaddsub
  SRC_clmcopy
  SRC_dipan
  SRC_dstart
  SRC_elast
  SRC_eosfit
  SRC_eosfit6
  SRC_filtvec
  SRC_fsgen
  SRC_initxspec
  SRC_irrep
  SRC_joint
  SRC_kgen
  SRC_kram
  SRC_lapw0
  SRC_lapw1
  SRC_lapw2
  SRC_lapw3
  SRC_lapw5
  SRC_lapw7
  SRC_lapwdm
  SRC_lapwso
  SRC_lcore
  SRC_lib
  SRC_lorentz
  SRC_lstart
  SRC_mini
  SRC_mixer
  SRC_nn
  SRC_optic
  SRC_optimize
  SRC_orb
  SRC_pairhess
  SRC_phonon
  SRC_qtl
  SRC_reformat
  SRC_sgroup
  SRC_spacegroup
  SRC_spaghetti
  SRC_structeditor
  SRC_sumpara
  SRC_supercell
  SRC_symmetry
  SRC_symmetso
  SRC_telnes3
  SRC_templates
  SRC_tetra
  SRC_trig
  SRC_txspec
  SRC_usersguide_html
  SRC_vecpratt
  example_struct_files
  TiC
  TiO2
  fccni
```

Thus, each program has its source code (split into several files) in its own subdirectory. All programs are written in FORTRAN90 (except SRC_sgroup and SRC_reformat, which are in C).

/SRC contains the users guide (in form of a postscript file **usersguide.ps** and as pdf-file **usersguide.pdf**), all c-shell scripts and some auxiliary files.

/SRC_usersguide_html contains the html version of the UG.

/Fccni, **/TiC** and **/TiO2** contain three example inputs and the respective outputs.

/example_struct_files contains a collection of various struct files, which could be of use especially for the less experienced user.

/SRC_templates contains various input templates.

In addition to the expansion of the tar-files **./expand_lapw** copies also all csh-shell scripts from **/SRC** to the current directory and creates links for some abbreviated commands.

11.2.2 Site configuration for WIEN2k

At the end of **expand_lapw** you will be prompted to start the script

./siteconfig_lapw

When you start this script for the first time (file **INSTALLDATE** not present), you will be guided through the setup process.

Later on you can use **siteconfig_lapw** to redimension parameters, update individual packages and recompile the respective programs.

During the first run, you will be asked to specify:

- your system; at this point system specific files (e.g. **cputim.f** will be installed. If your system is not listed, use the system **generic**, which should compile on any machine.
- your FORTRAN90 and C compilers;
- your compiler and linker options as well as the place for LAPACK and BLAS libraries. Depending on the system you selected, we have included some recommended compiler and linker options, which are known to work on our systems (use **generic** when you have problems here; see also sec. 11.2.4). On some systems it is required to specify LAPACK and BLAS libraries twice (i.e. R_LIBS:-llapack_lapw -lblas_lapw -llapack_lapw -lblas_lapw). This generates Makefiles from the corresponding Makefile.orig in all subdirectories.
- configuration of parallel execution will ask whether your system is shared memory, so that default parameters can be set accordingly (**$WIENROOT/parallel_options** is the file where this information is stored).
- to configure parallel execution for distributed systems, specify the command to open a remote shell, which on most systems is **rsh** or **ssh**.
- You will then be asked wether you want to run fine-grained parallel. This is only possible if FFTW, MPI and SCALAPACK (included in newer mkl-versions) are installed on your system and requires a fast network (100Mb/s is not enough) or a shared memory machine. It pays off only for bigger cases (matrixsize $>$ 5000).
- You should define NMATMAX, i.e. the maximum matrixsize (number of basisfunctions). This value should be adjusted according to the memory of your hardware. Rough estimates are:
  NMATMAX= 5000 ==> 256MB (real, i.e. with inversion symmetry)
  NMATMAX=10000 ==> 1GB (real) (==> cells with about 80-150 atoms/unitcell)
  If you choose it too large, **lapw1** will start to "page" leading to inacceptable performance or a crash. NMATMAX will be automatically recuced (by $\sqrt{2}$) for complex (without inversion) cases and increased by $\sqrt{NPE}$ for mpi-parallel cases.

- Now you are prompted to compile all programs (this will be done using **make**) and the executables are copied to the **$WIENROOT** directory. Compilation might take quite some time.
- During compilation watch for error messages on the screen. If there are errors, you may need to change into the corresponding **SRC_*** directory and examine file **compile.msg** for details.
  Common errors are wrong specification of compiler, linking or library options. In such cases, adopt the Makefile in this directory and recompile using **make**. Once you have proper options, correct them globally in **siteconfig_lapw** and recompile.

Later on you can use **siteconfig_lapw** to change parameters, options or to update a package.

11.2.3 User configuration

Each **WIEN2k** user should run the script **userconfig_lapw**. This will setup a proper environment.

The script **userconfig_lapw** will do the following for you:

- set a path to **WIEN2k** programs
- set the stacksize to "unlimited"
- add aliases
- add environment variables (**$WIENROOT**, **$SCRATCH**)

to your **~/.cshrc** or**~/.bashrc** file. Eventually you should also edit these files and set the **$LD_LIBRARY_PATH** variable (path where compiler-libs or blas-libraries are located).

Note: This will work only when the csh, tcsh or bash-shell is your login shell. Depending on your settings you may have to add similar lines also in your ***.login*** *file. If you are using a different login-shell, edit your startup files manually.*

11.2.4 Performance and special considerations

The script **siteconfig_lapw** is provided for general configuration and compilation of the **WIEN2k** package. When you call this script for the first time and follow the suggested answers, **WIEN2k** should run on your system (see 11.2.2).

The codes in the individual subdirectories **/SRC_program** are compiled using make. The file **Makefile** is generated during installation using **Makefile.orig** as template.

In some directories the source files ***.frc, *.F** and **param.inc_r/c** contain both, the real and complex (for systems without inversion symmetry) version of the code. You create the coresponding versions with **make** and **make complex,** respectively. (The ***.frc** and ***.F** files will then be preprocessed automatically).

The fine-grained parallel versions **lapw0_mpi; lapw1_mpi, lapw1c_mpi, lapw2_mpi, lapw2c_mpi** are created using **make para** (lapw0) and **make rp; make cp**.

For timing purposes a subroutine **CPUTIM** is used in several programs and specific routines for IBM-AIX, HP-UX, DEC-OSF1, SGI and SUN are available. On other systems **cputim_generic.c** should work.

On some HP systems you may encounter problems like: "stack growth failure". You may recompile with -K, reconfigure your Unix-kernel (with increased stack-size) or put large arrays in the respective program into COMMONS.

Most of the CPU time will be spent in **lapw1** and (to a smaller extent) in **lapw2 and lapw0**. Therefore we recommend to optimize the performance for these 3 programs:

- Find out which compiler options (**`man f90`**) make these programs run faster. You could specify a higher optimization (-O3), or specify a particular processor architecture (-qarch=pwr5 or -R10000, ...).
- Good performance depends on highly optimized BLAS (and much less on LAPACK) libraries. Whenever it is possible, replace the supplied libraries (**`SRC_lib/blas_lapw SRC_lib/lapack_lapw`**), by routines from your vendor (mkl for Intel or AMD processors, aclm for AMD, essl for IBM, sunperf for SUN, complib.sgimath on SGI, ...). If such libraries are not available use the **`GOTO`**-library (http://www.tacc.utexas.edu/tacc-projects/gotoblas2/) which is a very competitive alternative. (Eventually you may try to optimize them yourself using the "ATLAS" system (see http://math-atlas.sourceforge.net), but this is no longer recommended. We provide an "old" ATLAS-BLAS for a Pentium3 with **WIEN2k**.

11.2.5 Global dimensioning parameters

WIEN2k is written in Fortran 90 and all important arrays are allocated dynamically. The only important parameters left are NMATMAX and NUME, specifying the maximum matrixsize (should be adjusted to the memory of your hardware, see above) and the maximum number of eigenvalues (must be increased for unitcells with large number of electrons)

Some less important parameters are still present and described in chapter "dimensioning parameters" of the respective section in chapter 6.

We recommend to use **`siteconfig_lapw`** for redimensioning and recompilation. In order to work properly, the parameter XXXX in the respective **`param.inc`** files must obey the following syntax:

```
PARAMETER(XXXX= ....)
```

Note: between "(", XXXX and "=" there must be no space.

11.3 Installation and Configuration of w2web

11.3.1 General issues

w2web requires **`perl`**, which should be available on most systems. (If not contact your system administrator or install it yourself from the WWW)

When you start **w2web** for the first time on the computer where you want to execute **WIEN2k** (you may have to telnet, ssh,.. to this machine) with the command **`w2web [-p xxxx]`**, you will be asked for a username/password (I recommend you use the same as for your UNIX login).

You must also specify a "port" number (which can be changed the next time you start **w2web**). If the default port (7890) used to serve the interface is already in use by some other process, you will get the error message `w2web failed to bind port 7890 - port already in use!`. Then you will have to choose a different port number (between 1024 and 65536) . Please remember this port number, you need it when connecting to the **w2web** server.

*Note: Only user **root** can specify port numbers below 1024!*

Once **w2web**has been started, use your favorite WWW-browser to connect to **w2web**, specifying the correct portnumber, e.g. **`firefox http://hostname_where_w2web_runs:7890`**

On certain sites a firewall may block all high ports and one cannot connect to this machine. In these cases you can create a *ssh-tunnel* using the following commands:

At your "local_host" (the PC in front of you) connect to the "w2web_host" (where you started **w2web**) using

```
ssh -fNL 2000:w2web_host:7890 user@w2web_host
```

On your local host use a web browser and connect with: **`firefox http:127.0.0.1:2000`**.

Using "*Configuration* " you can further tailor the behaviour according to your wishes. In particular you can define new "execution types" to adjust to your queuing system.

For example the line

```
batch=batch < %f
```

defines an execution type "batch" using the UNIX batch command. (**w2web** collects its commands in a temporary script and you can access it using %f).

If you run on a machine with a queuing system (like loadleveler, sun-grid-engine, or pbs) you may define an "execution type"

```
qsub=cat %f > w2web-job;qsub-wienjob_lapw
```

The following scripts may serve as templates: **`qsub-wienjob_lapw`** in **`$WIENROOT`** needs a master-job-template **`qsub-job0_lapw`** and examples for loadleveler and SGE are provided in **`$WIENROOT`** (you may need to adapt them ! Other examples you can find on our FAQ-page on the web). Of course, with some small modifications you can define several "execution types" with eg. different number of processors or mpi vs. k-point parallel runs,....

w2web saves several variables in startup files which are in the **`(`**~/.w2web) directory.

11.3.2 How does w2web work?

w2web acts like a normal web-server - except that it runs on a "user level port" instead of the default http-port 80. It serves html-files and executes perl-scripts or executes system or user commands on the server host.

11.3.3 w2web-files in you home directory

w2web creates on the first start of w2web on host "hostname" the directory **`.w2web/hostname`** in your home directory with the following content:

- .w2web/hostname/conf
- .w2web/hostname/logs
- .w2web/hostname/sessions

11.3.4 The configuration file conf/w2web.conf

In this file various configuration parameters are stored by **w2web**. To restrict the access to certain IP addresses you can add lines like:

```
deny=*.*.*.*
allow=128.130.134.* 128.130.142.10
```

11.3.5 The password file conf/w2web.users

This file is created during the first run of **w2web**.

If you remove this file, the next start of **w2web**will activate the installation procedure again.

11.3.6 Using the https-protocol with w2web

In order to use the https-protocol the perl-library Net::SSLeay in addition to the OpenSSL package must be installed on your system. Both are freely available.

Then you must include a line with **ssl=1** in w2web.conf.

If you run **w2web**-server in ssl-mode you need a site certificate for your server. You may use the supplied certificate in **$WIENROOT/SRC_w2web/bin/w2web.pem** (copy this file to your conf-directory and set the keyfile= /.w2web/¡hostname¿/conf/w2web.pem line in your w2web.conf).

This certificate will not expire until 2015, but usually browsers will complain that they do not know the Certificate Authority who issued this certificate - if you don't like this message, you must buy a certificate from VeriSign, Thawte or a similar CA.

Of course you must connect to *https:* instead of *http:*, i.e. use:

netscape https://hostname_where_w2web_runs:7890.

11.4 Environment Variables

WIEN2k uses the following environment variables:

WIENROOT base directory where **WIEN2k** is installed
PDFREADER specifies program to read pdf files (acroread, xpdf,...)
SCRATCH directory where **case.vector** and **case.help??** are stored.
EDITOR path and name of your prefered editor
STRUCTEDIT_PATH path where the structeditor tool is located
OCTAVE_PATH path where the structeditor tool is located
OCTAVE_EXEC_PATH path where octave looks for executables (structeditor)
XCRYSDEN_TOPDIR if this variable is set **WIEN2k** will activate all interface extensions to XCrysDen.
USE_REMOTE [0|1] determines whether parallel jobs are run in background (on shared memory machines) or using rsh. It is overridden by settings in **$WIENROOT/parallel_options**
WIEN_GRANULARITY Default granularity for parallel execution. It is overridden by setting the granularity in the **.machines** file or in **$WIENROOT/parallel_options**
WIEN_EXTRAFINE if set, the residual k-points are spread one by one over the processors.

In addition on some systems variables like:

LD_LIBRARY_PATH path to libraries of compiler and math-libs
OMP_NUM_THREADS on multi-core machines for parallelization in certain libraries (mkl, goto)

12 Trouble shooting

In this chapter hints are given for solving some difficulties that have occurred frequently. This chapter is by no means complete and the authors would appreciate further suggestions which might be useful for other users. Beside the printed version of the users guide, an online pdf version is available using **help_lapw**. You can search for a specific keyword (use ^f keyword) and hopefully find some information.

> There is a mailing list for **WIEN2k** related questions. To subscribe to this list send mail to:
> `majordomo@theochem.tuwien.ac.at`
> with the text "subscribe wien". You will then automatically be added to the mailing list
> `wien@theochem.tuwien.ac.at`
> Please make use of this list!

If an error occurs in one of the SCF programs, a file program.error is created and an error message is printed into these files. The **run_lapw** script checks for these files and will automatically stop if a non-empty error file occurs.

Check the files **case.dayfile** (which is written by **init_lapw** and **run_lapw**), **:log** (where a history of all commands using **x** is given) and ***.error** for possible explanations.

In several places the dimensions are checked. The programs print a descriptive error message and stop.

case.outputnn: This file gives error messages if the atomic spheres overlap. But it should also be used to check the distances between the atoms and the coordination number (same distance). If inconsistencies exists, your **case.struct** file may contain an error. A check for overlapping spheres is also included in **mixer** and **lapw1**.

case.outputd: Possible stops or warnings are:

"NO SYMMETRY OPERATION FOUND IN ROTDEF": This indicates that in your **case.struct** file either the positions of equivalent atoms are not specified correctly (only positive coordinates allowed!!) or the symmetry operations are wrong.

case.output1: Possible stops or warnings are:

"NO ENERGY LIMITS FOUND IN SELECT": This indicates that E_{top} or E_{bottom} could not be found for some $u_l(r, E_l)$. Check your input if it happens in the zeroth iteration. Later, (usually in the second to sixth iteration) it may indicate that in your SCF cycle something went wrong and you are using a crazy potential. Usually it means that mixing of the charge densities was diverging and large charge fluctuations occured. Check previous charges for being physically reasonable (grep for labels :NTOxx :CTOxx :DIS :NEC01). Usually this happens when your input is not ok, or for very ill conditioned problems (very rare), or more likely, when "Ghostbands" appeared (or

some states were missing) because of bad energy parameters in **case.in1**. You will probably have to delete **case.broy*** and **case.scf**, rerun **x dstart** and then change some calculational parameters. These could be: fixing some energy parameter (modify both, **case.in1** and **case.in1_orig** or try the -in1orig switch if you have used -in1new); switch to a broadening method (TEMP with eg. 0.010 mRy); or increase the k-mesh (magnetic metals); or reduce the mixing parameter in **case.inm** slightly (eg. to 0.1). In very difficult (magnetic) cases a PRATT mixing with eg. 0.01 mixing might be helpful at the beginning of the scf cycle (but later switch to MSEC1 again) !

"STOP RDC_22": This indicates that the overlap matrix is not positive definite. This usually happens if your **case.struct** file has some error in the structure or if you have an (almost) linear dependent basis, which can happen for large RKMAX values and/or if you are using very different (extremely small and large) sphere radii R_{MT}.

"X EIGENVALUES BELOW THE ENERGY emin": This indicates that X eigenvalues were found below emin. Emin is set in **case.in1** (see sec. 7.3.3) or in **case.klist** generated by KGEN, see 6.3, 6.5). It may indicate that your value of emin is too high or the possibility of ghostbands, but it can also be intentional to truncate some of the low lying eigenvalues.

If you don't find enough eigenvalues (e.g.: in a cell with 4 oxygens you expect 4 oxygen s bands at roughly -1 Ry) check the energy window (given at the end of the first k-point in **case.in1** or in **case.klist**) and make sure your energy parameters are found by subroutine SELECT or set them by hand at a reasonable value.

case.output2: Possible stops or warnings are:

"CANNOT BE FOUND": This warning, which could produce a very long output file, indicates that some reciprocal K-vector would be requested (through the k-vector list of lapw1), but was not present in the list of the K generated in lapw2. You may have to increase the NWAV, and/or KMAXx parameters in lapw2 or increase GMAX in **case.in2**. The problems could also arise from wrong symmetry operations or a wrong structure in **case.struct**.

"QTL-B VALUE": If larger than a few percent, this indicates the appearance of ghost bands, which are discussed below in section 12.1.
The few percent message (e.g up to 10 %) does not indicate a ghost band, but can happen e.g. in narrow d-bands, where the linearization reaches its limits. In these cases one can add a local orbital to improve the flexibility of the basis set. (Put one energy near the top and the other near the bottom of the valence band, see section 7.3.3).

FERMI LEVEL not converged (or similar messages). This can have several reasons: i) Try a different Fermi-Method (change TETRA to GAUSS or TEMP in **case.in2**). ii) Count the number of eigenvalues in **case.output1** and compare it with the number of valence electrons. If there are too few eigenvalues, either increase EMAX in **case.klist** (from 1.5 to e.g. 2.5) or check if your scf cycle had large charge oszillations (see **SELECT** error above)

If the SCF cycle stops somewhere (especially in the first few iterations), it is quite possible, that the source of the error is actually in a previous part of the cycle or even in a previous (e.g. the zeroth) iteration. Check in the **case.scf** file previous charges, eigenvalues, ... whether they are **physically reasonable** (see **SELECT** error above).

12.1 Ghost bands

Approximate linear dependence of the basis set or the linearization of the energy dependence of the radial wave functions (see section 2.2) can lead to spurious eigenvalues, termed "ghost bands".

The first case may occur in a system which has atoms with very different atomic sphere radii. Suppose you calculate a hydroxide with very short O-H bonds so that you select small R_{MT} radii for O and H such as e.g. 1.0 and 0.6 a.u., respectively. The cation may be large and thus you could choose a large R_{MT} of e.g. 2.4 a.u. However, this gives a four time larger effective RKmax for the cation than for H, (e.g. 16.0 when you select RKmax=4.0 in **case.in1**). This enormous difference in the convergence may lead to unphysical eigenvalues. In such cases choose lmax=12 in **case.in1** (in order to get a very good re-expansion of the plane waves) and reduce R_{MT} for the cation to e.g. 1.8 a.u.

The second case can occur when you don't use a proper set of local orbitals. In this situation the energy region of interest (valence bands) falls about midway between two states with different principle quantum numbers, but with the same l-value (for one atom).

Take for example Ti with its $3p$ states being occupied as (semi-core) states, while the $4p$ states remain mostly unoccupied. In the valence band region neither of those two states (Ti $3p$, $4p$) should appear. If one uses 0.2 Ry for the expansion energy E(1) for the p states of Ti, then Ti-p states do appear as ghost bands. Such a run is shown below for TiO_2 (rutile).

The lowest six eigenvalues at GAMMA fall between about -1.30 and -1.28 Ry. They are ghost bands derived from fictitious Ti-p states. The next four eigenvalues between -0.94 and -0.78 Ry correspond to states derived from O 2s states, which are ok, since there are four O's per unit cell, four states are found.

The occurrence of such unphysical (indeed, unchemical!) ghostbands is the first warning that something went wrong. A more definite warning comes upon running LAPW2, where the corresponding charge densities are calculated. If the contribution to the charge density from the energy derivative of the basis function [the B_{lm} coefficient in equ. 2.4,2.7] is significant (i.e. much more than 5 per cent) then a warning is issued in LAPW2.

In the present example it reads:

```
     QTL-B VALUE .EQ. 40.35396 !!!!!!
```

This message is found in both the **case.scf** file and in **case.output2**.

When such a message appears, one can also look at the partial charges (QTL), which are printed under these conditions to OUTPUT2, and always appear in the files **case.helpXXX**, etc., where the last digit refers to the atomic index.

In the file below, note the E(1) energy parameter as well as the 6 ghost band energies around -1.29.

```
--------------- top of file:tio2.scf ----------------------------
          ATOMIC SPHERE DEPENDENT PARAMETERS FOR ATOM  Titanium
          OVERALL ENERGY PARAMETER IS     .2000
          E( 0)=     .2000
--->      E( 1)=     .2000
          E( 2)=     .2000   E(BOTTOM)=    -.140   E(TOP)= -200.000


          ATOMIC SPHERE DEPENDENT PARAMETERS FOR ATOM  Oxygen
          OVERALL ENERGY PARAMETER IS     .2000
          E( 0)=    -.7100   E(BOTTOM)=   -2.090   E(TOP)=     .670

       K=    .00000    .00000    .00000           1
:RKM  : MATRIX SIZE= 599  RKM= 6.99  WEIGHT= 8.00  PGR:
       EIGENVALUES ARE:
        -1.2970782   -1.2970782   -1.2948747   -1.2897193  -1.2897193
        -1.2882306    -.9389111    -.8484857    -.7880729   -.7880729
         -.0484830    -.0162982     .0121181     .0976534    .0976534
          .1914068     .1914068     .2341991     .3286919    .3477629
          .3477629     .3809219     .5143729     .5356211    .5550735
          .5617155     .5617155     .7087550     .7197110    .8736991
          .8736991     .9428865     .9533619    1.2224570   1.2224570
         1.4285169
       ******************************************************
       NUMBER OF K-POINTS:          1
```

```
:NOE  : NUMBER OF ELECTRONS          =  48.000
:FER  : F E R M I - ENERGY           =    .53562

:POS01: AT.NR. -1  POSITION =  .00000  .00000  .00000  MULTIPLICITY=  2
       LMMAX=10
       LM=  0 0 2 0 2 2 4 0 4 2 4 4 6 0 6 2 6 4 6 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0
:CHA01: TOTAL CHARGE INSIDE SPHERE   1 =     8.802166
:PCS01: PARTIAL CHARGES SPHERE =  1 S,P,D,F,PX,PY,PZ,D-Z2,D-X2Y2,D-XY,D-XZ,D-YZ
:QTL01:  .127 6.080 2.518  .067 2.011 2.047 2.022 1.090  .760 .155  .480  .034
                     VXX         VYY         VZZ       UP TO R
:VZZ01:            -4.96856     8.48379    -3.51524       2.000
:POS02: AT.NR. -2  POSITION =  .30500  .30500  .00000  MULTIPLICITY=  4
       LMMAX=16
       LM=  0 0 1 0 2 0 2 2 3 0 3 2 4 0 4 2 4 4 5 0 5 2 5 4 6 0 6 2 6 4 6 6 0 0
:CHA02: TOTAL CHARGE INSIDE SPHERE   2 =     5.486185
:PCS02: PARTIAL CHARGES SPHERE =  2 S,P,D,F,PX,PY,PZ,D-Z2,D-X2Y2,D-XY,D-XZ,D-YZ
:QTL02: 1.559 3.902  .022  .002 1.296 1.306 1.300  .014  .004 .000  .003  .001
                     VXX         VYY         VZZ       UP TO R
:VZZ02:             .25199     -.55091      .29892       1.600

:CHA  : TOTAL CHARGE INSIDE CELL =      48.000000
:SUM  : SUM OF EIGENVALUES =           -15.810906

   QTL-B VALUE .EQ.   40.35396   !!!!!!
      NBAND in QTL-file:         24
----------------end of truncated file tio2.scf----------------------
```

Next we show **`tio2.output2`** for the first of the ghost bands at -1.297 Ry. One sees that it corresponds mainly to a p-like charge, which originates from the energy derivative part Q(UE) of the Kohn-Sham orbital. Q(UE) contributes 40.1% compared with 8.5% from the main component Q(U). Q(UE) greater than Q(U) is a good indication for a ghost band.

```
---------------part of file tio2.output2 --------------------------
   QTL-B VALUE .EQ.   40.35396   !!!!!!
  K-POINT:   .0000   .0000   .0000   599  36          1
  BAND #  1  E= -1.29708  WEIGHT= 2.0000000
          L= 0     L= 1       PX:      PY:      PZ:    L= 2    DZ2:   DX2Y2:     DXY:     DXZ:     DYZ:    L= 3
 QINSID:    .0000  48.6035  35.0996  13.5039    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0030
 Q(U)  :    .0000   8.4902   6.0125   2.4777    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0026
 Q(UE) :    .0000  40.1132  29.0871  11.0261    .0000    .0000    .0000    .0000    .0000    .0000    .0000    .0005
          L= 0     L= 1       PX:      PY:      PZ:    L= 2      DZ2:   DX2Y2:     DXY:     DXZ:     DYZ:    L= 3
 QINSID:    .1294    .0707    .0000    .0055    .0653    .0088    .0038    .0049    .0000    .0000    .0000    .0022
 Q(U)  :    .1279    .0627    .0000    .0052    .0575    .0087    .0038    .0049    .0000    .0000    .0000    .0020
 Q(UE) :    .0016    .0081    .0000    .0003    .0077    .0001    .0000    .0000    .0000    .0000    .0000    .0002
 QOUT  : 1.9265
--------------------bottom of truncated file ---------------------
```

Another file in which the same information can be found is **`tio2.help031`**, since the ghost band is caused by a bad choice for the Ti-p energy parameter:

```
----------------------Top of file tio2.help031 ----------------------
  K-POINT:   .0000   .0000   .0000   599  36          1
  BAND #  1  E= -1.29708  WEIGHT= 2.0000000
  L= 0      .00000      .00000     .00000     .00000     .00000     .00000
  L= 1    48.60346     8.49022   40.11324     .00000     .00000     .00000
    PX:   35.09960     6.01247   29.08712     .00000     .00000     .00000
    PY:   13.50386     2.47774   11.02612     .00000     .00000     .00000
    PZ:     .00000      .00000     .00000     .00000     .00000     .00000
  L= 2      .00000      .00000     .00000     .00000     .00000     .00000
   DZ2:     .00000      .00000     .00000     .00000     .00000     .00000
 DX2Y2:     .00000      .00000     .00000     .00000     .00000     .00000
   DXY:     .00000      .00000     .00000     .00000     .00000     .00000
   DXZ:     .00000      .00000     .00000     .00000     .00000     .00000
   DYZ:     .00000      .00000     .00000     .00000     .00000     .00000
  L= 3      .00304      .00255     .00050     .00000     .00000     .00000
  L= 4      .00000      .00000     .00000     .00000     .00000     .00000
  L= 5      .00096      .00082     .00014     .00000     .00000     .00000
  L= 6      .00000      .00000     .00000     .00000     .00000     .00000
-------------------bottom of truncated file--------------------------
```

Note again for L=1 the percentage of charge associated with the primary (APW) basis functions ul (8.5%) versus that coming from the energy derivative component (40.1%).

If a ghost band appears, one should first analyze its origin as indicated above, then use appropriate local orbitals to improve the calculation and get rid of these unphysical states.

Do not perform calculations with "ghost-bands", even when the calculation converges.

Good luck !

13 References

Abt R., Ambrosch-Draxl C. and Knoll P. 1994 Physica B 194-196

Abt R. 1997 PhD Theses, Univ.Graz

Andersen O.K. 1973 Solid State Commun. 13, 133

— 1975 Phys. Rev. B 12, 3060

Ambrosch-Draxl C., Blaha P., and Schwarz K. 1991 Phys.Rev. B44, 5141

Ambrosch-Draxl C., Majewski J. A., Vogl P., and Leising G. 1995, PRB 51 9668

Ambrosch-Draxl C. and Sofo J., 2006 Comp. Phys. Comm. 175, 1

V.I. Anisimov, I.V. Solovyev, M.A. Korotin, M.T. Czyzyk, and G.A. Sawatzky, Phys. Rev. B **48**, 16929 (1993).

V.I. Anisimov, J. Zaanen, and O.K. Andersen, Phys. Rev. B **44**, 943 (1991)

Bader R. F. W. 2001: http://www.chemistry.mcmaster.ca/faculty/bader/aim/

Blaha P. and Schwarz K. 1983 Int. J. Quantum Chem. XXIII, 1535

Blaha P., Schwarz K., and Herzig P 1985 Phys. Rev. Lett. 54, 1192

Blaha P., Schwarz K., and Dederichs P 1988 Phys. Rev B 38, 9368

Blaha P., Schwarz K., Sorantin P.I. and Trickey S.B. 1990 Comp. Phys. Commun. 59, 399

Blaha P., Sorantin P.I., Schwarz K and Singh D. 1992 Phys. Rev. B 46, 1321

Blaha P., Hofstätter H., Koch R., Laskowski R. and Schwarz K. 2009, J.Comput.Phys. 229, 453.

Blöchl P.E., Jepsen O. and Andersen O.K. 1994, Phys. Rev B 49, 16223

Boettger J.C. and Albers R.C. 1989 Phys. Rev. B 39, 3010

Boettger J.C. and Trickey S.B. 1984 Phys. Rev. B 29, 6425

Brooks M.S.S. 1985 Physica B 130, 6

Charpin, T. 2001. (see **`$WIENROOT/SRC/elast-UG.ps`**)

Czyzyk M.T. and G.A. Sawatzky, Phys. Rev. B **49**, 14211 (1994).

Desclaux J.P. 1969 Comp. Phys. Commun. 1, 216; note that the actual code we use is an apparently unpublished relativistic version of the non-relativistic code described in this paper. Though this code is widely circulated, we have been unable to find a formal reference for it.

— 1975 Comp. Phys. Commun. 9, 31; this paper contains much of the Dirac-Fock treatment used in Desclaux's relativistic LSDA code.

O. Eriksson, B. Johansson, and M.S.S. Brooks, J. Phys. C **1**, 4005 (1989)

Feldman J.L., Mehl M.J., and Krakauer H. 1987 Phys. Rev. B 35, 6395

Gay David M., "ALGORITHM 611 – Subroutines for Unconstrained Minimization Using a Model/Trust-Region Approach", ACM Trans. Math. Software 9 (1983), pp. 503-524.

Hébert-Souche C., Louf P.-H., Blaha P., M. Nelhiebel, Luitz J., Schattschneider P., Schwarz K. and Jouffrey B.; The orientation dependent simulation of ELNES, Ultramicroscopy, 83, 9 (2000)

L.L. Hirst, Rev. Mod. Phys. **69**, 607 (1997)

Hohenberg P. and Kohn W. 1964 Phys. Rev. 136, B864

"International Tables for X-Ray Crystallography" 1964 Vol.1; The Kynoch Press, Birmingham UK

Jansen H.J.F. and Freeman A.J. 1984 Phys. Rev. B 30, 561

— 1986 Phys. Rev. B 33, 8629

Koelling D.D. 1972 J. Phys. Chem. Solids 33, 1335

Koelling D.D. and Arbman G.O. 1975 J.Phys. F: Met. Phys. 5, 2041

Koelling D.D. and Harmon B.N. 1977 J. Phys. C: Sol. St. Phys. 10, 3107

Kohler B., Wilke S., Scheffler M., Kouba R. and Ambrosch-Draxl C. 1996 Comp.Phys.Commun. 94, 31

Kohn W. and Sham L.J. 1965 Phys. Rev. 140, A1133

Kokalj A. 1999 J.Mol.Graphics and Modelling 17, 176

Krimmel H.G., Ehmann J., Elsässer C., Fähnle M. and Soler J.M. 1994, Phys.Rev. B50, 8846

Kuneš J, Novák P., Schmid R., Blaha P. and Schwarz K. 2001, Phys. Rev. B64, 153102

Kara, M. and Kurki-Suonio K. 1981 Acta Cryst A37, 201

Liberman D., Waber J.T., and Cromer D.T. 1965, Phys. Rev. 137A, 27

A.I. Liechtenstein, V. I. Anisimov, J. Zaanen, Phys. Rev. B **52**, R5467 (1995)

Luitz J., Maier M., Hébert C., Schattschneider P., Blaha P., Schwarz K., Jouffrey B. 2001 Eur. Phys J. B 21, 363-367

MacDonald A. H., Pickett, W. E. and Koelling, D. D. 1980 J. Phys. C 13, 2675

Madsen G. K. H., Blaha P, Schwarz K, Sjöstedt E and Nordström L 2001, Phys. Rev. B64, 195134

Marks L. D., and Luke R. 2008, Phys. Rev. B 78, 075114

Mattheiss L.F. and Hamann D.R. 1986 Phys. Rev. B 33, 823

Mattsson A., Armiento R., Paier J., Kresse G., Wills J. and Mattsson T 2008 J. Chem. Phys. 128, 084714

Meyer-ter-Vehn J. and Zittel W. 1988 Phys. Rev. B37, 8674

Moruzzi V.L., Janak J.F., and Williams A.R. 1978 "Calculated Properties of Metals" (Pergamon, NY)

Murnaghan F.D., Proc.Natl.Acad.Sci. USA 30, 244 (1944)

Neckel A., Schwarz K., Eibler R. and Rastl P. 1975 Microchim.Acta, Suppl.6, 257

Nelhiebel M., Louf P. H., Schattschneider P., Blaha P., Schwarz K. and Jouffrey B.; Theory of orientation sensitive near-edge fine structure core-level spectroscopy, Phys.Rev. B59, 12807 (1999)

Novak P. 1997 see **$WIENROOT/SRC/novak_lecture_on_spinorbit.ps**

Novák P. , Boucher F., Gressier P., Blaha P. and Schwarz K. 2001 Phys. Rev. B 63, 235114

Novák P. 2001 see **$WIENROOT/SRC/novak_lecture_on_ldaumatrixelements.ps** and **http://www.wien2k.at/reg_user/textbooks**

Novak P. 2006 see **$WIENROOT/SRC/Bhf_3.ps** and **http://www.wien2k.at/reg_user/textbooks**

Pardini L., Bellini V., Manghi F. and Ambrosch-Draxl C. 2011, Comp.Phys.Commun. subm. (http://arxiv.org/abs/1104.1558)

Perdew J.P, Chevary J.A., Vosko S.H., Jackson K.A., Pederson M.R., Singh D.J., and Fiolhais C. 1992 Phys.Rev.B46, 6671

Perdew J.P. and Wang Y. 1992, Phys.Rev. B45, 13244

Perdew J.P., Burke S. and Ernzerhof M. 1996, Phys.Rev.Let. 77, 3865

Perdew J.P., Kurth S., Zupan J. and Blaha P. 1999, Phys.Rev.Let. 82, 2544

Perdew J.P. et al. 2008, Phys. Rev. Let. 100, 136406

Pratt G.W. 1952 Phys. Rev. 88, 1217

Ray A.K. and Trickey S.B. 1981 Phys. Rev. B24, 1751; erratum 1983, Phys. Rev. B28, 7352

Rondinelli JM, Beng Bin and Marks LD. 2007, Comp. Mater. Sci. 40, 345-353 (also: Los Alamos archive, physics/0608160 (http://xxx.lanl.gov/abs/physics/0608160)

Schwarz K., Neckel A and Nordgren J, J.Phys.F:Metal Phys. 9, 2509 (1979)

Schwarz K., and Wimmer E, J.Phys.F:Metal Phys. 10, 1001 (1980)

Schwarz K. and Blaha P.: Lecture Notes in Chemistry 67, 139 (1996)

Schwarz K., P.Blaha and Madsen, G. K. H. Comp.Phys.Commun. 147, 71 (2002)

Singh D., Krakauer H., and Wang C.-S. 1986 Phys. Rev. B34, 8391

Singh, D. 1989 Phys. Rev. B40, 5428

Singh D. 1991, Phys.Rev. B43, 6388

Singh D. and Nordström L 2006, Plane waves, pseudopotentials and the LAPW method, 2^{nd} edition, Springer, New York

Sjöstedt E, Nordström L and Singh D. J. 2000 Solid State Commun. 114, 15

Sofo J and Fuhr J 2001: **$WIENROOT/SRC/aim_sofo_notes.ps**

Soler J.M. and Williams A.R. 1989, Phys.Rev. B40, 1560

Sorantin P.I., and Schwarz K.H. 1992, Inorg.Chem. 31, 567

Stahn J, Pietsch U, Blaha P and Schwarz K. 2001, Phys.Rev. B63, 165205

Tao Jianmin, Perdew J.P., Staroverov V. and Scuseria G. 2003, Phys.Rev.Let. 91, 146401

Tran F, Blaha P Schwarz K and Novak P 2006, Phys. Rev. B 74, 155108

Tran F, Laskowski R, Blaha P and Schwarz K. 2007, Phys. Rev. B 75, 115131

Tran F and Blaha P 2009, Phys. Rev. Lett. 102, 226401

von Barth U. and Hedin L. 1972 J. Phys. C.: Sol. St. Phys. 5, 1629

Wei S.H., Krakauer H., and Weinert M. 1985 Phys. Rev. B 32, 7792

Weinert M. 1981 J. Math. Phys. 22, 2433

Weinert M., Wimmer E., and Freeman A.J. 1982 Phys. Rev. B26, 4571

Wimmer E., Krakauer H., Weinert M., and Freeman A.J. 1981 Phys. Rev. B24, 864

Wu Z., Cohen R., 2006 Phys. Rev. B73, 235116

Yanchitsky B. and Timoshevskii T. 2001, Comp.Phys.Commun. 139, 235

Yu R., Singh D. and Krakauer H. 1991, Phys.Rev. B43, 6411

Part IV

Appendix

A Local rotation matrices

Local rotation matrices are used to rotate the global coordinate system (given by the definition of the Bravais matrix) to a local coordinate system for each atomic site. They are used in the program for two reasons:

- to minimize the number of LM combinations in the lattice harmonics expansion (of potential and charge density according to equ. 2.10). For example for point group mm2 one needs for L=1 just LM=1,0 if the coordinate system is chosen such that the z-axis coincides with the 2-fold rotation axis, while in an arbitrary coordinate system the three terms 1,0; 1,1 and -1,1 are needed (and so on for higher L).
- The interpretation e.g. of the partial charges requires a proper orientation of the coordinate system. In the example given above, the p orbitals split into 2 irreducible representations, but they can be attributed to p_z and p_x, p_y only if the z-axis is the 2-fold rotation axis.

It is of course possible to perform calculations without "local rotation matrices", but in such a case the LM combinations given in Table 7.42 (and by SYMMETRY) may not be correct. (The LM values for arbitrary orientations may be obtained from a procedure described in Singh 94.)

Fortunately, the "local rotation matrices" are usually fairly simple and are now **automatically inserted** into your **`case.struct`** file. Nevertheless we recommend to check them in order to be sure.

The most common coordinate transformations are

- interchanging of two axes (e.g. x and z)
- rotation by $45°$ (e.g. in the xy-plane)
- rotation of z into the (111) direction

Inspection of the output of SYMMETRY tells you if the local rotation matrix is the unit matrix or it gives you a clear indication how to find the proper matrix.

The local rotation matrix $\mathbf{R}$, which transforms the global coordinates r to the rotated ones r', is defined by $\mathbf{R}r = r'$.

There are two simple ways to check the local rotation matrixes together with the selected LM combinations:

- charge density plots generated with LAPW5 must be continuous across the atomic sphere boundary (especially valence or difference density plots are very sensitive, see 8.6)
- Perform a run of LAPW1 and LAPW2 using the GAMMA-point only (or a complete star of another k point). In such a case, "wrong" LM combinations must vanish. Note that the latter is true only in this case. For a k mesh in the IBZ "wrong" LM combinations do not vanish and thus must be omitted!!

A first example for "local rotation matrices" is given for the rutile TiO2, which has already been described as an example in section 10.3. Also two other examples will be given (see below).

A.1 Rutile (TiO_2)

Examine the output from symmetry. It should be obvious that you need local rotation matrices for both, Ti and O:

```
....
Titanium   operation #  1     1
Titanium   operation #  2     -1
Titanium   operation #  5     2 || z
Titanium   operation #  6     m n z
Titanium   operation # 12     m n 110
Titanium   operation # 13     m n -110
Titanium   operation # 18     2 || 110
Titanium   operation # 19     2 || -110
  pointgroup is mmm (neg. iatnr!!)
  axes should be: m n z, m n y, m n x
```

This output tells you, that for Ti a mirror plan normal to z is present, but the mirror planes normal to x and y are missing. Instead, they are normal to the (110) plane and thus you need to rotate x, y by 45° around the z axis. (The required choice of the coordinate system for mmm symmetry is also given in Table 7.42)

```
....
Oxygen     operation #  1     1
Oxygen     operation #  6     m n z
Oxygen     operation # 13     m n -110
Oxygen     operation # 18     2 || 110
  pointgroup is mm2 (neg. iatnr!!)
  axes should be: 2 || z, m n y
```

For O the 2-fold symmetry axes points into the (110) direction instead of z. The appropriate rotation matrices for Ti and O are:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \end{pmatrix}$$

A.2 Si Γ-phonon

Si possesses a face-centered cubic structure with two equivalent atoms per unit cell, at ($\pm$0.125, $\pm$0.125, $\pm$0.125). The site symmetry is -43m. For the Γ-phnon the two atoms are displaced in opposite direction along the (111) direction and cubic symmetry is lost. The output of SYMMETRY gives the following information:

```
Si         operation #  1     1
Si         operation # 13     m n -110
Si         operation # 16     m n -101
Si         operation # 17     m n 0-11
Si         operation # 24     3 || 111
Si         operation # 38     3 || 111
  pointgroup is 3m (neg. iatnr!!)
     axis should be: 3 || z, m n y
lm: 0 0  1 0  2 0  3 0  3 3  4 0  4 3  5 0  5 3  6 0  6 3  6
```

Therefore the required local rotation matrix should rotate z into the (111) direction and thus the matrix in the “struct“ file should be:

```
 0.4082483 -.7071068 0.5773503
 0.4082483 0.7071068 0.5773503
 -.8164966 0.0000000 0.5773503
```

$$\begin{matrix} \frac{\sqrt{6}}{6} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ -2\frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \end{matrix}$$

A.3 Trigonal Selenium

Selenium possesses space group P3121 with the following struct file:

```
H    LATTICE,NONEQUIV.ATOMS: 1
MODE OF CALC=RELA  POINTGROUP:32
 8.2500000 8.2500000  9.369000
ATOM= -1: X= .7746000  Y= .7746000  Z= 0.0000000
          MULT= 3          ISPLIT= 8
ATOM= -1: X= .2254000  Y= .0000000  Z= 0.3333333
ATOM= -1: X= .0000000  Y= .2254000  Z= 0.6666667
Se         NPT=  381  R0=.000100000 RMT=2.100000000  Z:34.0
LOCAL ROT.MATRIX:    0.0       0.5000000 0.8660254
                     0.0000000 -.8660254 0.5000000
                     1.0000000 0.0000000 0.0
   6    IORD OF GROUP G0
......
```

The output of SYMMETRY reads:

```
Se         operation #  1     1
Se         operation #  9     2 $|$$|$ 110
  pointgroup is 2 (neg. iatnr!!)
  axis should be: 2 || z
lm: 0 0  1 0  2 0  2 2 -2 2  3 0  3 2 -3 2  4 0  4 2 -4 2 ......
```

Point group 2 should have its 2-fold rotation axis along z, so the local rotation matrix can be constructed in two steps: firstly interchange x and z (that leads to z $\parallel$ (011)) and secondly rotate from (011) into (001) (see the struct file given above). Since this is a hexagonal lattice, SYMMETRY uses the hexagonal axes, but the local rotation matrix must be given in cartesian coordinates.

B Periodic Table

Periodic Table of the Elements

1**H** $1s^1$																		2**He** $1s^2$
3**Li** $He2s^1$	4**Be** $He2s^2$											5**B** $He2s^22p^1$	6**C** $He2s^22p^2$	7**N** $He2s^22p^3$	8**O** $He2s^22p^4$	9**F** $He2s^22p^5$	10**Ne** $He2s^22p^6$
11**Na** $Ne3s^1$	12**Mg** $Ne3s^2$											13**Al** $Ne3s^23p^1$	14**Si** $Ne3s^23p^2$	15**P** $Ne3s^23p^3$	16**S** $Ne3s^23p^4$	17**Cl** $Ne3s^23p^5$	18**Ar** $Ne3s^23p^6$
19**K** $Ar4s^1$	20**Ca** $Ar4s^2$	21**Sc** $Ar3d^14s^2$	22**Ti** $Ar3d^24s^2$	23**V** $Ar3d^34s^2$	24**Cr** $Ar3d^54s^1$	25**Mn** $Ar3d^54s^2$	26**Fe** $Ar3d^64s^2$	27**Co** $Ar3d^74s^2$	28**Ni** $Ar3d^84s^2$	29**Cu** $Ar3d^{10}4s^1$	30**Zn** $Ar3d^{10}4s^2$	31**Ga** $Ar3d^{10}4s^24p^1$	32**Ge** $Ar3d^{10}4s^24p^2$	33**As** $Ar3d^{10}4s^24p^3$	34**Se** $Ar3d^{10}4s^24p^4$	35**Br** $Ar3d^{10}4s^24p^5$	36**Kr** $Ar3d^{10}4s^24p^6$
37**Rb** $Kr5s^1$	38**Sr** $Kr5s^2$	39**Y** $Kr4d^15s^2$	40**Zr** $Kr4d^25s^2$	41**Nb** $Kr4d^45s^1$	42**Mo** $Kr4d^55s^1$	43**Tc** $Kr4d^55s^2$	44**Ru** $Kr4d^75s^1$	45**Rh** $Kr4d^85s^1$	46**Pd** $Kr4d^{10}$	47**Ag** $Kr4d^{10}5s^1$	48**Cd** $Kr4d^{10}5s^2$	49**In** $Kr4d^{10}5s^25p^1$	50**Sn** $Kr4d^{10}5s^25p^2$	51**Sb** $Kr4d^{10}5s^25p^3$	52**Te** $Kr3d^{10}5s^25p^4$	53**I** $Kr4d^{10}5s^25p^5$	54**Xe** $Kr4d^{10}5s^25p^6$
55**Cs** $Xe6s^1$	56**Ba** $Xe6s^2$	57-71 **La-Lu**	72**Hf** $Xe4f^{14}5d^26s^2$	73**Ta** $Xe4f^{14}5d^36s^2$	74**W** $Xe4f^{14}5d^46s^1$	75**Re** $Xe4f^{14}5d^56s^2$	76**Os** $Xe4f^{14}5d^66s^2$	77**Ir** $Xe4f^{14}5d^76s^2$	78**Pt** $Xe4f^{14}5d^96s^1$	79**Au** $Xe4f^{14}5d^{10}6s^1$	80**Hg** $Xe4f^{14}5d^{10}6s^2$	81**Tl** $Xe4f^{14}5d^{10}6s^26p^1$	82**Pb** $Xe4f^{14}5d^{10}6s^26p^2$	83**Bi** $Xe4f^{14}5d^{10}6s^26p^3$	84**Po** $Xe4f^{14}5d^{10}6s^26p^4$	85**At** $Xe4f^{14}5d^{10}6s^26p^5$	86**Rn** $Xe4f^{14}5d^{10}6s^26p^6$
87**Fr** $Rn7s^1$	88**Ra** $Rn7s^2$	89-103 **Ac-Lr**															

57**La** $Xe5d6s^2$	58**Ce** $Xe4f^26s^2$	59**Pr** $Xe4f^36s^2$	60**Nd** $Xe4f^46s^2$	61**Pm** $Xe4f^56s^2$	62**Sm** $Xe4f^66s^2$	63**Eu** $Xe4f^76s^2$	64**Gd** $Xe4f^75d6s^2$	65**Tb** $Xe4f^96s^2$	66**Dy** $Xe4f^{10}6s^2$	67**Ho** $Xe4f^{11}6s^2$	68**Er** $Xe4f^{12}6s^2$	69**Tm** $Xe4f^{13}6s^2$	70**Yb** $Xe4f^{14}6s^2$	71**Lu** $Xe4f^{14}5d6s^2$
89**Ac** $Rn6d7s^2$	90**Th** $Rn6d^27s^2$	91**Pa** $Rn5f^26d^17s^2$	92**U** $Rn5f^36d^17s^2$	93**Np** $Rn5f^46d^17s^2$	94**Pu** $Rn5f^67s^2$	95**Am** $Rn5f^77s^2$	96**Cm** $Rn5f^76d7s^2$	97**Bk** $Rn5f^97s^2$	98**Cf** $Rn5f^{10}7s^2$	99**Es** $Rn5f^{11}7s^2$	100**Fm** $Rn5f^{12}7s^2$	101**Md** $Rn5f^{13}7s^2$	102**No** $Rn5f^{14}7s^2$	103**Lr** $Rn5f^{14}6d7s^2$

WIEN2k
ISBN 3-9501031-1-2