WIEN2k

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

User’s Guide, WIEN2k_17.1 (Release 07/03/2017)

Peter Blaha
Karlheinz Schwarz
Georg Madsen
Dieter Kvasnicka
Joachim Luitz

Vienna University of Technology
Institute of Materials Chemistry
Getreidemarkt 9/165-TC
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,17.1 (Release 07/03/2017)

Prof. Dr. Karlheinz Schwarz
Vienna University of Technology
Institute of Materials Chemistry
Getreidemarkt 9/165-TC
A-1060 Vienna, Austria
ISBN 3-9501031-1-2

ISBN 3-9501031-1-2
Contents

1 Introduction to the WIEN2k package 1

1 Introduction 3

2 Basic concepts 7
  2.1 Density Functional Theory .......................... 7
  2.2 The APW Methods .................................. 8
    2.2.1 The LAPW Method .............................. 8
    2.2.2 The APW+lo Method ............................ 9
    2.2.3 General considerations ......................... 10

3 Quick Start 13
  3.1 Naming conventions ................................ 13
  3.2 Starting the server ................................ 14
  3.3 Connecting to the w2web server ..................... 15
  3.4 Creating a new session ............................. 15
  3.5 Creating a new case ............................... 16
  3.6 Creating the struct file ........................... 16
  3.7 Initialization ..................................... 18
  3.8 The SCF calculation ............................... 20
  3.9 The case.scf file ................................. 21
  3.10 Saving a calculation .............................. 21
  3.11 Calculating properties ............................ 21
    3.11.1 Electron density plots ......................... 21
    3.11.2 Density of States (DOS) ....................... 24
    3.11.3 X-ray spectra ................................ 26
    3.11.4 Bandstructure ................................ 26
    3.11.5 Bandstructure with band character plotting / full lines ....................... 27
    3.11.6 Volume Optimization ........................... 28
  3.12 Setting up a new case ............................ 29
    3.12.1 Manual setup ................................ 29
    3.12.2 Setting up a new case using w2web ............ 29
II Detailed description of the files and programs of the WIEN2k package

4 Files and Program Flow

4.1 Flow of input and output files ........................................ 33
4.2 Input/Output files .................................................. 37
4.3 The case.struct.file .................................................. 38
4.4 The case.scf file ................................................... 42
4.5 Flow of programs ..................................................... 43
  4.5.1 Core, semi-core and valence states ............................... 45
  4.5.2 Spin-polarized calculation ....................................... 45
  4.5.3 Fixed-spin-moment (FSM) calculations .......................... 45
  4.5.4 Antiferromagnetic (AFM) calculations .......................... 46
  4.5.5 Spin-orbit interaction .......................................... 46
  4.5.6 Orbital potentials .............................................. 47
  4.5.7 Onsite-exact-exchange and hybrid functionals for correlated electrons .......................... 48
  4.5.8 Unscreened and screened hybrid functionals (“hf”-module) .................. 50
  4.5.9 Slater, SmBJ and KLI potentials (“hf”-module) .................. 55
  4.5.10 modified Becke-Johnson potential (mBJ) for band gaps ............. 56
  4.5.11 DFT-D3 for dispersion energy .................................. 57
  4.5.12 Nonlocal van der Waals functionals ........................... 57

5 Shell scripts ..................................................................... 61

5.1 Job control ............................................................. 61
  5.1.1 Main execution script (x_lapw) .................................. 61
  5.1.2 Create the master input file case.struct (makestruct_lapw) ....... 63
  5.1.3 Job control for initialization (init_lapw) ......................... 63
  5.1.4 Job control for iteration (run_lapw or runsp_lapw) ............. 64
  5.1.5 Job control for iteration with the Slater/SmBJ/KLI potentials (run_vnonloc_lapw) ............. 66

5.2 Utility scripts .......................................................... 67
  5.2.1 Save a calculation (save_lapw) .................................. 67
  5.2.2 Restoring a calculation (restore_lapw) ......................... 68
  5.2.3 Reduce atomic spheres and interpolate density (reduce_rmt_lapw) ............. 68
  5.2.4 Remove unnecessary files (clean_lapw) ......................... 68
  5.2.5 Migrate a case to/from a remote computer (migrate_lapw) .......... 68
  5.2.6 Generate case.inst (instgen_lapw) ............................ 69
  5.2.7 Set R-MT values in your case.struct file (setrmt_lapw) ............ 69
  5.2.8 create_add_atom_clmsum_lapw .................................. 69
  5.2.9 Create case.int file (for DOS) (configure_int_lapw) ............. 70
5.8.3 Born effective charges ............................................ 96
5.8.4 Piezoelectric constants ........................................ 97
5.9 Getting on-line help .................................................. 97
5.10 Interface scripts .................................................... 98
  5.10.1 eplot\_lapw ................................................... 98
  5.10.2 gibbs\_lapw ................................................... 98
  5.10.3 parabolfit\_lapw ............................................. 99
  5.10.4 dosplot\_lapw ................................................ 99
  5.10.5 dosplot2\_lapw ............................................. 99
  5.10.6 Cgrace\_lapw, Cgrace\_conf\_lapw and Cgrace\_dos\_lapw ..... 100
  5.10.7 Curve\_lapw .................................................. 100
  5.10.8 specplot\_lapw .............................................. 100
  5.10.9 rhoplot\_lapw ............................................... 100
  5.10.10 prepare\_xsf\_lapw ........................................ 101
  5.10.11 opticplot\_lapw ........................................... 102
  5.10.12 addjoint-updn\_lapw ..................................... 102

6 Initialization .................................................... 103
  6.1 NN ................................................................. 103
    6.1.1 Execution .................................................. 104
  6.2 SGROUP .......................................................... 104
    6.2.1 Execution .................................................. 104
  6.3 SYMMETRY ....................................................... 104
    6.3.1 Execution .................................................. 105
  6.4 LSTART .......................................................... 105
    6.4.1 Execution .................................................. 105
    6.4.2 Dimensioning parameters .................................. 105
    6.4.3 Input ....................................................... 106
  6.5 KGEN ............................................................. 107
    6.5.1 Execution .................................................. 108
    6.5.2 Dimensioning parameters .................................. 108
  6.6 DSTART .......................................................... 108
    6.6.1 Execution .................................................. 108
    6.6.2 Dimensioning parameters .................................. 109
7 SCF cycle

7.1 LAPW0
  7.1.1 Execution
  7.1.2 Dimensioning parameters
  7.1.3 Input

7.2 DFTD3
  7.2.1 Execution
  7.2.2 Input

7.3 NLVDW
  7.3.1 Execution
  7.3.2 Input

7.4 ORB
  7.4.1 Execution
  7.4.2 Dimensioning parameters
  7.4.3 Input

7.5 LAPW1
  7.5.1 Execution
  7.5.2 Dimensioning parameters
  7.5.3 Input

7.6 HF
  7.6.1 Execution
  7.6.2 Input

7.7 LAPWSO
  7.7.1 Execution
  7.7.2 Dimensioning parameters
  7.7.3 Input

7.8 LAPW2
  7.8.1 Execution
  7.8.2 Dimensioning parameters
  7.8.3 Input

7.9 SUMPARA
  7.9.1 Execution
  7.9.2 Dimensioning parameters

7.10 LAPWDM
  7.10.1 Execution
  7.10.2 Dimensioning parameters
  7.10.3 Input

7.11 LCORE
8.11.2 Dimensioning parameters ............................................. 165
8.11.3 Input ............................................................................. 165
8.12 LAPW3 ............................................................................. 166
  8.12.1 Execution ..................................................................... 166
  8.12.2 Dimensioning parameters ............................................. 166
8.13 LAPW5 ............................................................................. 166
  8.13.1 Execution ..................................................................... 167
  8.13.2 Dimensioning parameters ............................................. 167
  8.13.3 Input ............................................................................. 167
8.14 LAPW7 ............................................................................. 169
  8.14.1 Execution ..................................................................... 170
  8.14.2 Dimensioning parameters ............................................. 170
  8.14.3 Input ............................................................................. 171
8.15 MINI ................................................................................. 172
  8.15.1 Execution ..................................................................... 173
  8.15.2 Dimensioning parameters ............................................. 173
  8.15.3 Input ............................................................................. 173
8.16 NMR ................................................................................. 175
8.17 OPTIC ................................................................................. 175
  8.17.1 Execution ..................................................................... 177
  8.17.2 Dimensioning parameters ............................................. 177
  8.17.3 Input ............................................................................. 178
8.18 OPTIMIZE ......................................................................... 179
  8.18.1 Execution ..................................................................... 179
  8.18.2 Input ............................................................................. 179
8.19 QTL ................................................................................. 180
  8.19.1 Execution ..................................................................... 180
  8.19.2 Input ............................................................................. 180
  8.19.3 Output ......................................................................... 182
8.20 SPAGHETTI ........................................................................ 182
  8.20.1 Execution ..................................................................... 183
  8.20.2 Input ............................................................................. 183
8.21 TELNES3 ........................................................................... 185
  8.21.1 Execution ..................................................................... 186
  8.21.2 Input ............................................................................. 186
  8.21.3 Practical considerations .............................................. 191
  8.21.4 Files .............................................................................. 191
8.22 TETRA .............................................................................. 192
9.21 Tmaker .............................................................. 210
9.22 struct2cif .......................................................... 210
9.23 struct2poscar ....................................................... 210
9.24 conv2prim .......................................................... 210
9.25 fleur2wien .......................................................... 210
9.26 StructGen of w2web .............................................. 211
9.27 supercell ........................................................... 211
  9.27.1 Execution ....................................................... 211
9.28 structeditor ......................................................... 211
  9.28.1 Execution ....................................................... 213
9.29 Visualization ....................................................... 213
  9.29.1 BALSAC ......................................................... 213
  9.29.2 XCrysDen ....................................................... 214
9.30 Unsupported software ............................................ 214

10 Examples .......................................................... 215
  10.1 TiC ................................................................. 215
  10.2 FCC Nickel ........................................................ 215
  10.3 Rutile ............................................................... 216
  10.4 supercell calc ..................................................... 217
  10.5 Further examples ................................................. 218

III Installation of the WIEN2k package and Dimensioning of programs 219

11 Installation and Dimensioning ..................................... 221
  11.1 Requirements ..................................................... 221
    11.1.1 Installation tips for mpich, fftw (either version 2.1.5 or 3.3), and ELPA 222
    11.1.2 Installation of LIBXC ........................................ 223
  11.2 Installation of WIEN2k .......................................... 223
    11.2.1 Expanding the WIEN2k distribution ....................... 223
    11.2.2 Site configuration for WIEN2k ............................. 225
    11.2.3 User configuration ........................................... 226
    11.2.4 Performance and special considerations .................. 226
    11.2.5 Global dimensioning parameters .......................... 227
  11.3 w2web ............................................................ 227
    11.3.1 General issues .............................................. 227
    11.3.2 How does w2web work? ..................................... 228
    11.3.3 w2web-files in you home directory ....................... 228
    11.3.4 The configuration file conf/w2web.conf ................... 228
11.3.5 The password file conf/w2web.users ........................................... 228
11.3.6 Using the https-protocol with w2web ........................................ 228
11.4 Environment Variables ................................................................. 229

12 Trouble shooting .............................................................................. 231
12.1 Ghost bands .................................................................................. 232

IV Appendix .......................................................................................... 237

A Local rotation matrices ......................................................................... 239
A.1 Rutile (TiO$_2$) ................................................................................. 240
A.2 Si Γ-phonon .................................................................................... 240
A.3 Trigonal Selenium ........................................................................... 241

B Periodic Table .................................................................................... 243
## List of Tables

4.1 Input and output files of init programs .......................... 35
4.2 Input and output files of utility programs .......................... 36
4.3 Input and output files of main programs in an SCF cycle ............ 37
4.4 Lattice type, description and bravais matrix used in WEN2k. The angle $\gamma'$ is defined via $\cos(\gamma) = \cos(\gamma') \sin(\alpha) \sin(\beta) + \cos(\beta) \cos(\alpha)$ .......................... 39
6.6 Relativistic quantum numbers ................................ 107
7.3 global XC-switches .................................................. 116
7.4 EX-switches .......................................................... 116
7.5 EC-switches .......................................................... 117
7.6 VX-switches .......................................................... 118
7.7 VC-switches .......................................................... 119
7.54 LM combinations of “Cubic groups” (3∥(111)) direction, requires “positive atomic index” in case.struct. Terms that should be combined [Kara and Kurki-Suonio, 1981] must follow one another .......................... 136
7.55 LM combination and local coordinate system of “non-cubic groups” (requires “negative atomic index” in case.struct) .......................... 137
8.93 Possible values of QSPLIT and their interpretation .................. 181
8.116 Quantum numbers of the core state involved in the x-ray spectra .... 197
## List of Figures

2.1 Partitioning of the unit cell into atomic spheres (I) and an interstitial region (II) ..... 8

3.1 TiC in the sodium chloride structure. This plot was generated using BALSAC (see 9.29.1). Interface programs between \textbf{WIEN2k} and BALSAC are available. ..... 14

3.2 Startup screen of \textbf{w2web} ................................................................. 15

3.3 Main window of \textbf{w2web} ................................................................. 16

3.4 \textbf{StructGen} of \textbf{w2web} ............................................................... 17

3.5 List of input files ............................................................... 20

3.6 Task “Electron Density Plots” .......................................................... 22

3.7 Electron density of TiC in (100) plane using Xcrysden .......................... 23

3.8 Electron density of TiC in (100) plane ................................................ 24

3.9 Density of states of TiC ............................................................... 25

3.10 Density of states of TiC ............................................................... 25

3.11 Ti $L_{III}$ spectrum of TiC ............................................................... 26

3.12 Bandstructure of TiC ............................................................... 27

3.13 Bandstructure of TiC, showing t$_{2g}$-character bands of Ti in character plotting mode. 28

3.14 Energy vs. volume curve for TiC ....................................................... 28

4.1 Data flow during a SCF cycle (programX.def, case.struct, case.inX, case.outputX and optional files are omitted) ........................................ 34

4.2 Program flow in \textbf{WIEN2k} ........................................................... 44

5.1 Flow chart of \textbf{lapw1para} .............................................................. 84

5.2 Flow chart of \textbf{lapw2para} .............................................................. 85

7.1 Schematic dependence of DOS and $u_l(r, E_l)$ on the energy ............... 127

9.1 3D electron density in TiC generated with XCrysDen ....................... 214
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
Institute of Materials Chemistry
Getreidemarkt 9/165-TC
A-1060 Vienna, Austria
Fax: +43-1-58801-165982

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:


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.
Part I

Introduction to the \texttt{WIEN2k} package
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 parallelization 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.

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
- M. Jamal (Iran) various scripts, 2D-optimize, IRelast, Tmaker
- K. Jorissen (Univ. Antwerp), C. Hebert (TU Wien), telnes3
- R. Laskowski (TU Vienna), structeditor, main developer of mpi-parallelization, new dstart version, NMR module
- E. Kabliman (TU Vienna), arrows
- F. Karsai (TU Vienna), elast, lapwso
- D. Koller (TU Vienna), prepare_xsf_lapw
- 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, lapwdm, qtl, dipan
- 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), various xc-functionals, Forces for orbital potential, Hybrid-Functionals
- P. Wissgott (TU Vienna) join_vectorfiles
- 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. Chizmeshya (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:

\[
\text{pblaha@theochem.tuwien.ac.at} \\
\text{kschwarz@theochem.tuwien.ac.at}
\]
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, 1964], [Kohn and Sham, 1965]). Therein the key quantities are the spin densities \( \rho_\sigma(r) \) in terms of which the total energy is

\[
E_{\text{tot}}(\rho_\uparrow, \rho_\downarrow) = T(\rho_\uparrow, \rho_\downarrow) + E_{\text{ee}}(\rho_\uparrow, \rho_\downarrow) + E_{\text{Ne}}(\rho_\uparrow, \rho_\downarrow) + E_{\text{xc}}(\rho_\uparrow, \rho_\downarrow) + E_{\text{NN}}
\]

with \( E_{\text{NN}} \) 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 \( E_{\text{xc}} \) can be written in terms of a local exchange-correlation energy density \( \mu_{\text{xc}} \) times the total (spin-up plus spin-down) electron density as

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

and ii), the particular form chosen for that \( \mu_{\text{xc}} \). 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 [Perdew and Wang, 1992]. \( E_{\text{tot}} \) has a variational equivalent with the familiar Rayleigh-Ritz principle. The most effective way known to minimize \( E_{\text{tot}} \) by means of the variational principle is to introduce orbitals \( \chi_{ik}^\sigma \) constrained to construct the spin densities as

\[
\rho_\sigma(r) = \sum_{i,k} \rho_{ik}^\sigma |\chi_{ik}^\sigma(r)|^2 \tag{2.2}
\]

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

\[
[-\nabla^2 + V_{\text{Ne}} + V_{\text{ee}} + V_{\text{xc}}^\sigma] \chi_{ik}^\sigma(r) = \epsilon_{ik}^\sigma \chi_{ik}^\sigma(r) \tag{2.3}
\]

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.
CHAPTER 2. BASIC CONCEPTS

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. PW91 [Perdew et al., 1992] or Perdew, Burke and Ernzerhof (PBE) [Perdew et al., 1996], which is the recommended option.

Recent meta-GGA versions called PKZB [Perdew et al., 1999] and TPSS [Tao et al., 2003] employ 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., 2002].

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, 1977] or with the second variational method including spin-orbit coupling ([MacDonald et al., 1980], [Novák, 1997]). Core states are treated fully relativistically [Desclaux, 1969].

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, 1973, Andersen, 1975], [Koelling, 1972], [Koelling and Arbman, 1975] [Wimmer et al., 1981],[Weinert, 1981], [Weinert et al., 1982], [Blaha and Schwarz, 1983], [Blaha et al., 1985], [Wei et al., 1985], [Mattheis and Hamann, 1986], [Jansen and Freeman, 1984], [Schwarz and Blaha, 1996]. An excellent book by [Singh and Nordström, 2006] 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](image)

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)

$$\phi_{kn} = \sum_{lm} [A_{lm, kn}, u_l(r, E_l) + B_{lm, kn}, \dot{u}_l(r, E_l)] Y_{lm}(\hat{r})$$  \hspace{1cm} (2.4)

where $u_l(r, E_l)$ is the (at the origin) regular solution of the radial Schrödinger 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 Schrödinger equation on a radial mesh inside the sphere.

2. (II) in the interstitial region a plane wave expansion is used

$$\phi_{kn} = \frac{1}{\sqrt{\omega_e}} e^{i k_n \cdot r}$$  \hspace{1cm} (2.5)

where $k_n = k + K_n$; $K_n$ are the reciprocal lattice vectors and $k$ 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

$$\psi_k = \sum_n c_n \phi_{kn}$$  \hspace{1cm} (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 3$s$ and 4$s$ energy) and one energy derivative (at one of these energies):

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

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

### 2.2.2 The APW+lo method

Sjöstedt, Nordström and Singh [Sjöstedt et al., 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:

$$\phi_{k_n} = \sum_{lm} [A_{lm,k_n} u_l(r, E_l)] Y_{lm}(\hat{r})$$  \hspace{1cm} (2.8)

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

$$V(r) = \begin{cases} \sum_{LM} V_{LM}(r) Y_{LM}(\tilde{r}) & \text{inside sphere} \\ \sum_K V_K e^{i K \cdot \tilde{r}} & \text{outside sphere} \end{cases}$$  \hspace{1cm} (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., 1982]. 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., 1991]. For the implementation of this formalism in WIEN see [Kohler et al., 1996] and [Madsen et al., 2001]. An alternative formulation by [Soler and Williams, 1989] has also been tested and found to be equivalent, both in computationally efficiency and numerical accuracy [Krimmel et al., 1994].

The Fermi energy and the weights of each band state can be calculated using a modified tetrahedron method [Blöchl et al., 1994], 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 et al., 1980], [Singh and Nordström, 2006] and [Novák, 1997]. 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., 2001]).

PROPERTIES:

The density of states (DOS) can be calculated using the modified tetrahedron method of [Blöchl et al., 1994].

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), see [Neckel et al., 1975], [Schwarz et al., 1979, Schwarz and Wimmer, 1980]

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-Draxl et al., 1995, Abt et al., 1994, Abt, 1997] and in particular [Ambrosch-Draexl and Sofo, 2006]. 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 [Sofo and Fuhr, 2001].
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:
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.

\[ \text{netscape http://hostname.where.w2web.runs:7890} \]

(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](image-url)
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 then 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](image)

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):

<table>
<thead>
<tr>
<th>Title</th>
<th>TiC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice</td>
<td>F (for face centered)</td>
</tr>
<tr>
<td>a</td>
<td>4.328 Å (make sure the Ang button is selected)</td>
</tr>
<tr>
<td>b</td>
<td>4.328 Å</td>
</tr>
<tr>
<td>c</td>
<td>4.328 Å</td>
</tr>
<tr>
<td>α, β, γ</td>
<td>90</td>
</tr>
<tr>
<td>Atom</td>
<td>Ti, enter position (0,0,0)</td>
</tr>
<tr>
<td>Atom</td>
<td>C, enter position (5,5,5)</td>
</tr>
</tbody>
</table>

Click “Save Structure” (Z will be updated automatically) and “set automatically RMT and continue editing”:
3.6. CREATING THE STRUCT FILE

This will compute the nearest neighbor 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/rmt.html](http://www.wien2k.at/reg_user/faq/rmt.html). 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.

![StructGen](image)

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.
CHAPTER 3. QUICK START

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 \[
\text{show all files}^
\]”, 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 basic input file has been created, initialization of the calculation is done by “Execution \[
\text{initialize calc.}^
\]”. For structures given by experiment (not man-made supercells,...) you would usually run in “Fast mode, just specify a few most important imput parameters (or use the defaults).

However, this introduction will guide you through all individual 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 5.1.3.

Alternatively you could run the script init_lapw [-b] 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 Utials. \[
\text{show dayfile}^
\].

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)

\(\text{view TiC.outputnn} :\) 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

\(\text{view TiC.outputs}group \) : 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).

\(\text{view TiC.outputs} :\) 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.
3.7. INITIALIZATION

*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 (e.g., 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.in0st`, `in1st`, `in2st`, `incst` and `inmst`. 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.in1st*: 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 (3 if you have small H-spheres) are usually reasonable. Eventually you could 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).

*edit TiC.in2st*: Here you may limit (increase the LM expansion, increase the value of GMAX (in cases with small spheres (e.g., systems with H-atoms) it will be automatically increased anyway) 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.inic`, `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” calculation needs 10 times as much). 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 a “Fast-mode”, which is the recommended default and where the most important 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!! Only for hand-made `case.struct` files (e.g., using supercells, ...) one should run the first steps (from `nn` to `symmetry` step by step, since in such cases these programs may rewrite `case.struct` and specify different multiplicities or even change the unit cell.

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).
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 5.1.4.

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.
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 buttons, 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
3.11. CALCULATING PROPERTIES

ORTHO

For a detailed description of input options consult section 8.13.3
- Calculate electron density (x lapw5)
- Plot output (using rhoplot), after the first preview select a range $z_{\text{min}}=-0.5$ to $z_{\text{max}}=2$

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).
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 -qt1). (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 8.22.3
- 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,...)
3.11. CALCULATING PROPERTIES

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 (\texttt{x lapw2 -qt1}). This is only possible, when you still have a valid \texttt{TiC.vector} file on a tetrahedral mesh. To reproduce this figure you will have to increase the EMAX value in your \texttt{TiC.inl} to 2.5 Ry and rerun \texttt{x lapw1}
- Edit \texttt{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 8.23.3.
- Calculate spectra
- Preview spectra

If you want to use the supplied plotting interface \texttt{specplot} to preview the results, the public domain program \texttt{gnuplot} must be installed on your computer. The calculated TiC \(L_{III}\)-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 \texttt{TiC.klist.band} from the template in \texttt{SWIENROOT/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 \texttt{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 \texttt{lapw1.def} such that the k-mesh is read from \texttt{TiC.klist.band} and not from \texttt{TiC.klist})

\textbf{Note:} When you want to calculate DOS, charge densities or spectra after this bandstructure, you must first recalculate the \texttt{TiC.vector} file using the “tetrahedral” k-mesh, because the k-mesh for the band structure plots is \textbf{not} suitable for calculations of such properties.
3.11. CALCULATING 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](image)

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. 9.13). 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 by hand, but using some utilities. We recommend the script makestruct_lapw which will ask for the required input and create init.struct, which you should copy to case.struct. Alternatively, you can use cif2struct or xyz2struct to convert a “cif”, “txt” or “xyz” file into the WIEN2k case.struct file. Check page 209 for more info on the specific file formats.

You can also use the struct file from a similar case as pattern, but note, that the automatic setting of proper R0-values is not guaranteed by that procedure and you should use it only for VERY SIMILAR cases (elements). 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 209 for more info on the specific file formats.

For more information on the StructGen refer to page 211.
Part II

Detailed description of the files and programs of the WIEN2k package
4 File structure and program flow

Contents

4.1 Flow of input and output files ............................................. 33
4.2 Input/Output files .......................................................... 37
4.3 The case.struct.file ......................................................... 38
4.4 The case.scf file ............................................................. 42
4.5 Flow of programs ........................................................... 43

(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 \texttt{nn}, \texttt{sgroup}, \texttt{symmetry}, \texttt{lstart}, \texttt{kgen}, \texttt{dstart} (table 4.1), the utility programs \texttt{tetra}, \texttt{irrep}, \texttt{spaghetti}, \texttt{aim}, \texttt{lapw7}, \texttt{elnes}, \texttt{lapw3}, \texttt{lapw5}, \texttt{xspec}, \texttt{optic}, \texttt{joint}, \texttt{kram}, \texttt{optimize} and \texttt{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 \texttt{programX.def} files. The data flow is illustrated in Fig. 4.1.

<table>
<thead>
<tr>
<th>program</th>
<th>needs</th>
<th>generates</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>\texttt{nn.def}</td>
<td>\texttt{case.outputnn}</td>
</tr>
<tr>
<td>SGROUP</td>
<td>\texttt{case.struct}</td>
<td>\texttt{case.outputsgroup}</td>
</tr>
<tr>
<td>SYMMETRY</td>
<td>\texttt{synergy.def}</td>
<td>\texttt{case.in2.st}</td>
</tr>
<tr>
<td>LSTART</td>
<td>\texttt{lstart.def} \texttt{case.struct} \texttt{case.in1.st}</td>
<td>\texttt{case.outputlstart} \texttt{case.rsp} \texttt{case.in1.st} \texttt{case.in2.st} \texttt{case.in1.in2} \texttt{case.in2.in} \texttt{case.inm restart}</td>
</tr>
<tr>
<td>KGEN</td>
<td>\texttt{kgen.def} \texttt{case.struct}</td>
<td>\texttt{case.outputkgen} \texttt{case.klist} \texttt{case.kgen}</td>
</tr>
<tr>
<td>DSTART</td>
<td>\texttt{dstart.def} \texttt{case.struct} \texttt{case.rsp(up)} \texttt{case.in1} \texttt{case.in1} \texttt{case.in2}</td>
<td>\texttt{case.outputdstart} \texttt{case.clmsum(up)} \texttt{dstart.error} \texttt{case.in0_std}</td>
</tr>
</tbody>
</table>

Table 4.1: Input and output files of init programs

<table>
<thead>
<tr>
<th>program</th>
<th>needs</th>
<th>generates</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPAGHETTI</td>
<td>\texttt{spaghetti.def} \texttt{case.insp} \texttt{case.struct} \texttt{case.output1}</td>
<td>\texttt{case.spaghetti_p} \texttt{case.outputsp} \texttt{case.output1} \texttt{case.band.agr}</td>
</tr>
<tr>
<td>TETRA</td>
<td>\texttt{tetra.def} \texttt{case.int} \texttt{case.kgen}</td>
<td>\texttt{case.outputt} \texttt{case.energy} \texttt{case.sigma}</td>
</tr>
<tr>
<td>LAPW3</td>
<td>\texttt{lapw3.def} \texttt{case.struct} \texttt{case.in2} \texttt{case.clmsum}</td>
<td>\texttt{case.output3} \texttt{case.rho} \texttt{case.clmsum}</td>
</tr>
<tr>
<td>LAPW5</td>
<td>\texttt{lapw5.def} \texttt{case.struct} \texttt{case.in5} \texttt{case.clmsumval}</td>
<td>\texttt{case.sigma} \texttt{case.output5} \texttt{case.rho} \texttt{case.rho.oned}</td>
</tr>
<tr>
<td>XSPEC</td>
<td>\texttt{xspec.def} \texttt{case.inc} \texttt{case.int} \texttt{case.vsp} \texttt{case.struct} \texttt{case.qtl}</td>
<td>\texttt{case.outputx} \texttt{case.dos1} \texttt{case.dos2} \texttt{case.xspec} \texttt{case.mspspec} \texttt{case.m1} \texttt{case.m2}</td>
</tr>
<tr>
<td>OPTIL</td>
<td>\texttt{optic.def} \texttt{case.struct} \texttt{case.mat.diag} \texttt{case.inop} \texttt{case.vsp} \texttt{case.vector}</td>
<td>\texttt{case.outputop} \texttt{case.symmat} \texttt{case.symmat1} \texttt{case.symmat2}</td>
</tr>
<tr>
<td>JOINT</td>
<td>\texttt{joint.def} \texttt{case.injoint}</td>
<td>\texttt{case.outputjoint} \texttt{case.joint} \texttt{case.sigma_intra} \texttt{case.intra}</td>
</tr>
</tbody>
</table>

continued on next page
### Table 4.2: Input and output files of utility programs

<table>
<thead>
<tr>
<th>program</th>
<th>needs</th>
<th>generates</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPW0</td>
<td>necessary</td>
<td>optional</td>
</tr>
<tr>
<td>case.struct</td>
<td>case.kgen</td>
<td>case.weight</td>
</tr>
<tr>
<td>KRAM</td>
<td>kram.def</td>
<td>case.inkram</td>
</tr>
<tr>
<td>OPTIMIZE</td>
<td>case.struct</td>
<td>case_initial.struct</td>
</tr>
<tr>
<td>MINI</td>
<td>mini.def</td>
<td>case.inM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>case.constraint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRREP</td>
<td>case.struct</td>
<td>case.vector</td>
</tr>
<tr>
<td>AIM</td>
<td>case.struct</td>
<td>case.climsum</td>
</tr>
<tr>
<td>LAPW7</td>
<td>case.struct</td>
<td>case.vector</td>
</tr>
<tr>
<td>QTL</td>
<td>case.struct</td>
<td>case.vector</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAPW1</td>
<td>case.struct</td>
<td>case.in1</td>
</tr>
<tr>
<td>LAPWS0</td>
<td>lapwso.def</td>
<td>case.struct</td>
</tr>
<tr>
<td>LAPW2</td>
<td>lapw2.def</td>
<td>case.struct</td>
</tr>
<tr>
<td>LAPWDM</td>
<td>lapwdm.def</td>
<td>case.struct</td>
</tr>
</tbody>
</table>

continued on next page
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 lapw).

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

- **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 8.13). 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 8.13). In spin-polarized calculations two files case.clmvalup and case.clmvaldn are used instead.

- **case.dmatup/dn** Contains the density matrix generated by lapw2 or lapwdm for LDA+U, OP or onsite-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.energy** Contains the eigenvalues of all k-points calculated in lapw1. In spin-polarized calculations two files case.energyup and case.energydn are used instead. lapwso generates case.energyso.

- **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 7.8.3) and in tetra.

- **case.klist** This file contains a list of k-points in the BZ on a (special k-point) tetrahedral mesh. It is generated 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 $r^2 \ast V_{LM}(r)$ 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.symmat Contains the momentum matrix elements between bands i,j. Created by optic and used in joint.

case.vcoul Contains the Coulomb potential (in the lattice harmonics representation as $r^2 \ast V_{LM}(r)$ 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 onsite-hybrid-DFT calculations in form of a $(2l+1,2l+1)$ matrix.

case.vtotal Contains the total potential (in the lattice harmonics representation as $r^2 \ast V_{LM}(r)$ 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.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 $r \ast V$ (thus the first values should be close to $-2 \ast Z$). 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, or the makestruct_lapw utility, you don’t have to bother with this file directly, but generate it by specifying the relevant data in a mask. Alternatively, the utilities cif2struct or xyz2struct convert the corresponding cif or xyz files to the WIEN2k-format.

However, the description of the fields of this master input file 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! Also some parameters are usually element-specifically chosen (R0)
We start the description of this file with an abridged example for rutile TiO\(_2\) (adding line numbers):

```
--------------------- top of file ---------------------

Titaniumdioxide TiO\(_2\) (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 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 11
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 12
LOCAL ROT MATRIX: 0.0000000 -.7071068 0.7071068 13
0.0000000 0.7071068 0.7071068 14
1.0000000 0.0000000 0.0000000 15
16 SYMMETRY OPERATIONS:
1 1 0 0.00 16
0 1 0 0.00
0 0 1 0.00
1 0 0 0.00
1 1 0.00
1 1 0.00
0 0 1.00
1 1 0.00
0 1 0.00
0 1 0.00
0 0 1.00
1 1 0.00
1 1 0.00
2 1 0.00
1 1 0.00
1 1 0.00
1 1 0.50
1 0 0.50
1 0 0.50
1 0 0.50
1 1 0.50
16

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

Interpretive comments on this file are as follows.

Table 4.4: Lattice type, description and bravais matrix used in WIEN2k. The angle \(\gamma'\) is defined via

\[
\cos(\gamma) = \cos(\gamma') \sin(\alpha) \sin(\beta) + \cos(\beta) \cos(\alpha)
\]

```

| P | all primitive lattices except hexagonal |
| F | face-centered |
| B | body-centered |
| CY | A-base-centered (orthorhombic only) |
| CZX | B-base-centered (orthorhombic only) |
| R | rhombohedral |

| lattice type | as defined in table 4.4. For centered monoclinic lattices only the CXZ setting is supported and the monoclinic angle must be gamma. Eventually you have to transform a given spacegroup setting into one supported in WIEN2k (for instance for SG #12 we need (B112/m) or (B2/m11) setting and not (C122/m1) or (C2/m11) (it depends on your starting setting, but the final setting must have a monoclinic angle gamma); for SG #15 we need (B2/b) or one of the alternative B settings, but not one of the many others) using the Bilbao crystallographic server (http://www.cryst.ehu.es/; “structure utilities”; SETSTRU) |

| NAT | number of inequivalent atoms in the unit cell |
```

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

```
RELAX FULLY RELATIVISTIC CORE AND SCALAR RELATIVISTIC VALENCE
NREL NON-RELATIVISTIC CALCULATION

[a, b, c, α, β, γ]

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:

\[
a_{\text{hex}} = 2\cos\left(\frac{\pi - \alpha_{\text{rhomb}}}{2}\right)a_{\text{rhomb}}
\]

\[
c_{\text{hex}} = 3\sqrt{a_{\text{rhomb}}^2 - \frac{1}{4}a_{\text{hex}}^2}
\]

AND (FOR FCC-LIKE LATTICES)

\[
a_{\text{rhomb}} = a_{\text{cubic}}/\sqrt{2}
\]

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

[atom-index, x, y, z]

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 ≤ x ≤ 1; the positions in the unit cell are consistent with the convention used in the International Tables of Crystallography [Tab, 1964]. IN FACE- (BODY-) CENTERED STRUCTURES ONLY ONE OF FOUR (TWO) ATOMS MUST BE GIVEN, E.G. 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:

\[
\begin{bmatrix}
0 & 1 & 0 \\
\frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}} & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{3}} \\
-1 & 1 & 0 \\
1 & 1 & 1
\end{bmatrix}
\]

[isplit]

NUMBER OF EQUIVALENT ATOMS OF THIS KIND
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
4.3. THE CASE: STRUCTURE

-2  d-z2, d-x2y2, d-xy, (d-xz,d-yz)
88  split lm like charges (for old teines, not necessary anymore)
99  calculate cross-terms (for old teines, not necessary anymore)

>>> line 5 must now be repeated MULT-1 times for the other positions of each equivalent atom according to the Wyckoff position in [Tab, 1964].

line 7: format (A10,5X,I5,5X,F10.8,5X,F10.5,5X,F10.5)
   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: $r(n) = R_0 \times e^{(n-1)\cdot DX}$
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 5.2.7). 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 7.8.3 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 [Tab, 1964])
   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 [Tab, 1964])
   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
After a line "Precise positions", a list of all atomic positions can follow with full machine precision. These coordinates are written by mixer if one performs a “MSR1a” structure optimization and they will be used instead of the truncated numbers read above (only if they “agree”, but not if one modifies them by hand such that they differ more significantly).

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). If there is a “WARNING” mentioned, check :WAR
- :WAR contains some warnings indicating that there might be a problem with your calculations. Usually these problems are not fatal, but may influence the accuracy.
- :DIS charge distance between last 2 iterations (∫ |ρn − ρn−1|dr). Good convergence criterium.
- :FER Fermi energy (and Fermi-method)
- :GAP energy gap (for insulators). Please note, this value will only be correct, if the VBM/CBM are in your k-mesh. (“Shifted” k-meshes do not contain the Gamma-point and often gaps are at Gamma !!)
- :FORxx force on atom xx in mRy/bohr (in the local (for each atom) cartesian 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))
- :FR in MSR1a mode prints information about the remaining size of the forces and whether it will/has switched to MSR1 mode.
- :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)
- :ETLxx partial charges in sphere xx
- :EPLxx partial charges and “mean energies” in lower (semicore) energy window for atom xx. Used as energy parameters in case.in1 for next iteration
- :EHXX partial charges and “mean energies” in higher (valence) energy window for atom xx. Used as energy parameters in case.in1 for next iteration
- :EFGLxx 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)
- :MKM 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
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):

- **SETRMT**: a perl-program which helps to select proper RMT values
- **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 useful 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.
Figure 4.2: Program flow in \textit{WIEN2k}
4.5. FLOW OF PROGRAMS

Then a self-consistency cycle is initiated and repeated until convergence criteria are met (see 3.8 and 5.1.4). 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.

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 equilibrium). 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.vectorordn 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 lapwl -up (see also the comments for iterative diagonalization in section 5.2.22).

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 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 [Kuneš 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 temporarily 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 useful 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 useful 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 in `lapw2` (or in `lapwdm` when spin-orbit is also used), it will be mixed in `mixer` (consistently with the “regular” charge density) and the orbital dependent 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_vc_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 Onsite-exact-exchange and hybrid functionals for correlated electrons

In `WIEN2k`, it is also possible to go beyond standard LDA (GGA) and include onsite-exact-exchange (i.e., Hartree-Fock), which is very useful for strongly correlated systems, since such calculations are computationally nearly as cheap as standard DFT (or LDA+U). The onsite-exact-exchange/hybrid methods apply HF only inside the atomic spheres and only to one particular orbital. Thus you can use it only for localized electrons (see [Tran et al., 2006] for details). **Onsite-exact-exchange** will NOT improve gaps in sp-semiconductors. For these systems you have to use full hybrid-DFT (see Sec.4.5.8) or the mBJ potential (see Sec.4.5.10).

**Onsite hybrid functionals for correlated electrons:**

The one-parameter onsite hybrid functionals have the general following form:

\[
E_{\text{onsite-hybrid}}^{xc}[\rho] = E_{\text{SL}}^{xc}[\rho] + \alpha \left( E_{x}^{HF}[\Psi_{\text{corr}}] - E_{x}^{SL}[\rho_{\text{corr}}] \right)
\]

where \( E_{\text{SL}}^{xc} \) is the underlying semilocal (SL) functional. The following semilocal functionals can be used in \( E_{\text{onsite-hybrid}}^{xc} \):

- **LDA**: XC_LDA in `case.in0`. mode=HYBR and fraction=\( \alpha \) in `case.ineece`.
- **PBE**: XC_PBE in `case.in0`. mode=HYBR and fraction=\( \alpha \) in `case.ineece`.
- **WC**: XC_WC in `case.in0`. mode=HYBR and fraction=\( \alpha \) in `case.ineece`.
- **PBEsol**: XC_PBEsol in `case.in0`. mode=HYBR and fraction=\( \alpha \) in `case.ineece`.
- **TPSS**: XC_TPSS in `case.in0`. mode=HYBR and fraction=\( \alpha \) in `case.ineece`.
- **REVTPSS**: XC_REVTPSS in `case.in0`. mode=HYBR and fraction=\( \alpha \) in `case.ineece`.
- **MBJ**: XC_MBJ in `case.in0`. mode=HYBR and fraction=\( \alpha \) in `case.ineece`. Please note, that the double counting correction will utilize LDA (instead of MBJ).

The three-parameter onsite hybrid functionals B3PW91 and B3LYP are also available. These two functionals were proposed with the fraction of exact exchange \( \alpha = 0.2 \), however other values for \( \alpha \) can be chosen as well.

- **B3PW91**: EX_B3PW91 EC_B3PW91 VX_B3PW91 VC_B3PW91 in `case.in0`. mode = HYBR and fraction = 0.2 in `case.ineece`.

\[
E_{\text{onsite-B3PW91}}^{xc}[\rho] = E_{\text{LDA}}^{xc}[\rho] + 0.2 \left( E_{x}^{HF}[\Psi_{\text{corr}}] - E_{x}^{LDA}[\rho_{\text{corr}}] \right) + 0.72 \left( E_{x}^{B88}[\rho] - E_{x}^{LDA}[\rho] \right) + 0.81 \left( E_{c}^{PW92}[\rho] - E_{c}^{LDA}[\rho] \right)
\]

where \( E_{c}^{LDA} = E_{c}^{PW92} \).

- **B3LYP**: XC_B3LYP in `case.in0`. mode = HYBR and fraction = 0.2 in `case.ineece`.

\[
E_{\text{onsite-B3LYP}}^{xc}[\rho] = E_{\text{LDA}}^{xc}[\rho] + 0.2 \left( E_{x}^{HF}[\Psi_{\text{corr}}] - E_{x}^{LDA}[\rho_{\text{corr}}] \right) + 0.72 \left( E_{x}^{B88}[\rho] - E_{x}^{LDA}[\rho] \right) + 0.81 \left( E_{c}^{LYP}[\rho] - E_{c}^{LDA}[\rho] \right)
\]

where \( E_{c}^{LDA} = E_{c}^{VWN5} \).
Onsite exact-exchange functionals for correlated electrons:

Onsite Hartree-Fock calculations, i.e.,

\[ E_{\text{onsite}}^{\text{HF}}[\rho] = E_{\text{SL}}^{\text{xc}}[\rho] + E_{\text{HF}}^{\text{xc}}[\Psi_{\text{corr}}] - E_{\text{SL}}^{\text{xc}}[\rho_{\text{corr}}] \]

are also possible with the following semilocal functionals.

- **LDA**: XC$_{\text{LDA}}$ in case.in0. mode=EECE and fraction=1 in case.ineece
- **PBE**: XC$_{\text{PBE}}$ in case.in0. mode=EECE and fraction=1 in case.ineece
- **WC**: XC$_{\text{WC}}$ in case.in0. mode=EECE and fraction=1 in case.ineece
- **PBEsol**: XC$_{\text{PBEsol}}$ in case.in0. mode=EECE and fraction=1 in case.ineece
- **TPSS**: XC$_{\text{TPSS}}$ in case.in0. mode=EECE and fraction=1 in case.ineece
- **REVTPSS**: XC$_{\text{REVTPSS}}$ in case.in0. mode=EECE and fraction=1 in case.ineece
- **MBJ**: XC$_{\text{MBJ}}$ in case.in0. mode=EECE and fraction=\(\alpha\) in case.ineece. Please note, that the double counting correction will utilize LDA (instead of MBJ).

**Input and execution:**

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, natom
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
  - natom: 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)

**2nd line repeated nlorb-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)

The electron density of the correlated electrons (4f or 3d) could have a very sharp angular nodal structure and common GGAs behave very badly for such densities (check :FIT in **case.scf** or look up \( V_{xc}(R_{\text{mt}}) \) in **case.output0**. It is therefore often recommended to use LDA instead of a GGA for the double counting correction of the exchange term for the correlated electrons. This is possible by copying **case.in0** to **case.in0eece.lda** and specify VX_LDA (see 7.1 in the first line). Please note, for MBJ this will be done automatically and a warning will be issued.

▶ **runsp_lapw -eece [-so]** (or **runsp_c_lapw -eece**)

As with LDA+U, hybrid functionals can be used only for spin-polarized calculations and the switch **-eece** activates it. **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 dependent potentials.

### 4.5.8 Unscreened and screened hybrid functionals (“hf”-module)

The onsite exact-exchange/hybrid functionals from 4.5.7 can be applied only to localized electrons (typically 3d or 4f), but lead to cheap calculations. In **WIEN2k**, it is also possible to apply hybrid (and Hartree-Fock) functionals to all electrons, however this leads to calculations which are one or two orders of magnitude more expensive. Hybrid functionals are usually more accurate than the semilocal functionals for the electronic properties of semiconductors and insulators. They also give accurate results for strongly correlated systems like NiO. In hybrid functionals a fraction \( \alpha \) of semilocal (SL) exchange is replaced by Hartree-Fock (HF) exchange:

\[
E_{xc}^{\text{hybrid}} = E_{xc}^{\text{SL}} + \alpha \left( E_{x}^{\text{HF}} - E_{x}^{\text{SL}} \right)
\]

Hybrid functionals can also be constructed by considering only the short-range part of \( E_{x}^{\text{HF}} \) and \( E_{x}^{\text{SL}} \), which leads to the so-called “screened” hybrid functionals. In **WIEN2k**, the unscreened and screened hybrid functionals are implemented using the second-variational procedure [Tran and Blaha, 2011]. A few important points should be noted:

▶ Both the k-point and MPI parallelizations can be used (simultaneously or only one of them). As usual, the k-point parallelization is over the k-points in the irreducible Brillouin zone and is managed by c-shell scripts. There are two modes of MPI parallelization that can be used. In the first one (**mode1**), which is the default, these are the loops over the occupied bands in the main subroutine that are parallelized, and this should be the most efficient mode in terms of speed. However, if your system is too large in terms of memory requirement for your nodes, then you can try the second mode (**mode2**), which is much more efficient in terms of memory, but may eventually lead to slower calculations. With **mode2**, the first part of the main subroutine is parallelized over the matrix elements of the second variational Hamiltonian (roughly \( \text{nband}^2 \text{nband}/2 \) where \( \text{nband} \) is specified in **case.inhf**, see 7.6.1), while in the second part this is done for the loops over the occupied bands.
Due to the orbital-dependency of the HF potential, the files `case.vectorhf`, `case.energyhf` and `case.weighthf` are also saved when `save_lapw` is executed. If you restart a calculation without `case.vectorhf`, then, for the first iteration, it will be generated from the semilocal potential (lapw1), and therefore the number of scf iterations to reach convergence will be larger.

It is not possible to use spin-orbit coupling (-so) with hybrid functionals.

The available functionals

Among the semilocal functionals $E_{xc}^{SL}$ available in WIEN2k, only a few of them can be used in $E_{xc}^{hybrid}$ (both in unscreened and screened modes). The functionals are the following (`case.in0_grr` is for the exchange part $E_{xc}^{SL}$):

- **LDA**: XC_LDA in `case.in0` and EX_SLDA VX_SLDA in `case.in0_grr`
- **PBE**: XC_PBE in `case.in0` and EX_SPBE VX_SPBE in `case.in0_grr`. The functionals PBE0 and YS-PBE0 (similar to HSE06 [Heyd et al., 2003, Krukau et al., 2006]) correspond to $\alpha = 0.25$ in `case.inhf` (see 7.6).
- **WC**: XC_WC in `case.in0` and EX_SWC VX_SWC in `case.in0_grr`
- **PBEsol**: XC_PBEsol in `case.in0` and EX_SPBESOL VX_SPBESOL in `case.in0_grr`
- **BPW91** (this is not B3PW91): EX_B88 VX_B88 EC_PW91 VC_PW91 in `case.in0` and EX_SB88 VX_SB88 in `case.in0_grr`.
- **BLYP** (this is not B3LYP): EX_B88 VX_B88 EC_LYP VC_LYP in `case.in0` and EX_SB88 VX_SB88 in `case.in0_grr`.

For the screened hybrid functionals using PBE, PBEsol or B88 for exchange, an alternative for the screened exchange specified in `case.in0_grr` is to use the one proposed in [Henderson et al., 2008, Weintraub et al., 2009], HJS, which is based on the error function (instead of the Yukawa function) for the screening:

- **EX_SHJSPBE VX_SHJSPBE**
- **EX_SHJSPBESOL VX_SHJSPBESOL**
- **EX_SHJSB88 VX_SHJSB88**

Note that the screening parameter $\omega$ in HJS is set to $(2/3) \lambda$ of the value specified in `case.inhf` (see [Tran and Blaha, 2011]). Note also that the screening in the HF exchange is still done with the Yukawa function (with $\lambda$ for the screening) since the error-function-based screening in HF is not implemented.

In addition, calculations with the well-known B3PW91 and B3LYP (with VWN5) unscreened hybrid functionals (see 4.5.7 for the functionals form) can also be done:

- **B3PW91**: XC_B3PW91 in `case.in0`, EX_SLDA VX_SLDA in `case.in0_grr`, and select no screening and $\alpha = 0.2$ in `case.inhf`
- **B3LYP**: XC_B3LYP in `case.in0`, EX_SLDA VX_SLDA in `case.in0_grr`, and select no screening and $\alpha = 0.2$ in `case.inhf`

Hartree-Fock calculations (without correlation) are also possible:

- **HF**: EX_LDA VX_LDA in `case.in0`, EX_SLDA VX_SLDA in `case.in0_grr`, and select no screening and $\alpha = 1$ in `case.inhf`
Flow in \texttt{run\_lapw} -hf

The flow of programs during a scf iteration when executing \texttt{run\_lapw} -hf is the following (non-spin-polarized and real case):

\begin{itemize}
  \item x \texttt{lapw0 -grr} (semilocal exchange)
  \item x \texttt{lapw0} (semilocal exchange-correlation)
  \item x \texttt{lapw1} (semilocal orbitals)
  \item x \texttt{lapw2} (semilocal bands)
  \item \texttt{mv case\_vectorhf case\_vectorhf\_old}
  \item x \texttt{hf} (hybrid orbitals)
  \item x \texttt{lapw2 -hf} (hybrid electron density and bands)
  \item x \texttt{lcore}
  \item x \texttt{mixer}
\end{itemize}

Self-consistent calculation

The steps to perform a calculation with hybrid functionals are the following:

\begin{itemize}
  \item Do a calculation with the underlying semilocal functional $E_{xc}^{\text{SL}}$ (recommended but not mandatory).
  \item "save" the semilocal calculation.
\end{itemize}

The next steps can be done conveniently using \texttt{init\_hf\_lapw [-up]}:

\begin{itemize}
  \item create \texttt{case\_inhf} (cp $\text{WIENROOT}/\text{SRC\_templates/template\_inhf case\_inhf}). While there are (more or less) reasonable default values for most parameters, you must set \texttt{nband} manually. Set \texttt{nband} to at least the number of occupied bands plus one. Larger values are more accurate, but be aware that computing time scales as $n_{\text{band}}^2$. The value of \texttt{nband} which leads to a converged result will strongly depend on the studied system and property (e.g., much higher for EFG than for band gap or lattice constant).
  \item create \texttt{case\_in0\_grr} (cp \texttt{case\_in0 case\_in0\_grr}), this file contains:
    \begin{itemize}
      \item a screened exchange functional (e.g., EX\_SPBE VX\_SPBE) for the exchange functional (see above)
      \item "R2V" option (instead of "NR2V") such that the exchange potential is written to \texttt{case\_r2v\_grr}
      \item "KXC" (instead of "TOT") such that $\alpha E_{xc}^{\text{SL}}$ is printed in \texttt{case\_scf0\_grr} (and \texttt{case\_scf}) under the label :AEXSL
    \end{itemize}
  \item In \texttt{case\_inc} the print switch has to be "1" for all atoms such that the core orbitals are printed in \texttt{case\_corewf} (you don’t have to set this manually, the script \texttt{run\_lapw} will do it automatically when \texttt{-hf} is specified).
  \item if \texttt{nband} is large, you may have to edit \texttt{case\_in1} and set EMAX to a higher value (e.g., 5 Ry).
  \item Execute \texttt{run\_kgenhf\_lapw}. This generates \texttt{case\_klist\_fbz, case\_klist\_ibz, case\_kgen\_ibz} and \texttt{case\_outputkgenhf}. One must use identical k-meshes and shifts for IBZ and FBZ. Note that the k-parallelization is done over the k-points specified in \texttt{case\_klist\_ibz} (irreducible Brillouin zone).
\end{itemize}

All these steps above can be conveniently performed using the script \texttt{init\_hf\_lapw}.

Once the initializations has been done, execute \texttt{run\_lapw} (or \texttt{runsp\_lapw}) with the switch \texttt{-hf}:

\texttt{run\_lapw -hf}
Neglect of the nondiagonal terms

If only the eigenvalues are wanted, you may use the switch `-diaghf`. By using this switch, only the diagonal elements of the 2nd variational Hamiltonian matrix are calculated (the non-diagonal elements are set to zero). This leads to a much faster calculation of the eigenvalues, while keeping a very good accuracy [Tran, 2012]. However, the orbitals will not be modified, therefore running the calculation for more than one iteration is useless (the result will not change except for metallic systems). This option is not recommended for systems which are described as metallic with the semilocal functional or for difficult systems (e.g., NiO, see [Tran, 2012]). It is important to be aware that with this option, the total energy (:ENE in `case.scf`) is wrong unless the option `-nonself` (see below) is also used. After having done and saved a well converged calculation with the semilocal functional, the setting up of such a calculation is the same as for a self-consistent calculation (see above), but then `run(sp).lapw` is executed with `-diaghf` (`-hf` and `-i 1` will be set automatically):

```
run(sp).lapw -diaghf ...
```

The option `-diaghf` can be used simultaneously with the options `-nonself` and `-redklist`.

Reduced k-mesh for the HF potential

In order to reduce the computational time for the calculation of the HF potential, the internal loop over the k-points can be reduced to a subset of k-points. For instance, for a calculation with a $12 \times 12 \times 12$ k-mesh, the reduced k-mesh for the HF potential can be one of the following: $6 \times 6 \times 6$, $4 \times 4 \times 4$, $3 \times 3 \times 3$, $2 \times 2 \times 2$ or $1 \times 1 \times 1$. This option, which should be used only in the case of screened exchange (see [Paier et al., 2006]), is particularly interesting for total energy calculations. Obviously, choosing such a reduced k-mesh is an approximation which needs to be tested. The setting up of such a calculation is the same as for a self-consistent calculation (see above), but with the switch `-redklist` when executing `run_kgenhf_lapw` (to create also `case.klist_rfbz`, `case.klist_ribz` and `case.kgen_ribz`):

```
run_kgenhf_lapw -redklist
```

and `run(sp).lapw`:

```
run(sp).lapw -hf -redklist ...
```

The option `-redklist` can be used simultaneously with the options `-nonself` and `-diaghf`.

Non-self-consistent calculation of the total energy for hybrid functionals

It is possible to calculate the total energy non-self-consistently, i.e., by plugging the orbitals obtained from a calculation with the underlying semilocal functional $E_{xc}^{SL}$ into the total-energy hybrid functional. By doing so, the 2nd variational Hamiltonian is not calculated and therefore the computational time will be reduced significantly. After having done and saved a well converged calculation with the semilocal functional, the setting up of a non-self-consistent calculation is the same as for a self-consistent calculation (see above), but with the additional switch `-nonself` (`-hf` and `-i 1` will be set automatically) that has to be used in `run(sp).lapw`:

```
run(sp).lapw -nonself ...
```

This option can be particularly interesting for the calculation of the lattice constant, which depends mainly on the functional, but very little on the orbitals plugged into the functional. It can be used simultaneously with the option `-diaghf` and `-redklist`. 
Starting a calculation from another k-mesh

Due to the orbital-dependence of the HF potential, it is not straightforward to start directly a calculation with a potential generated from a previous calculation with another k-mesh. However, due to the high cost of a hybrid calculation, it is desirable to have this possibility in order to reduce the number of iterations during the scf procedure.

This option is also useful if a vector file on a very dense k-mesh is needed, e.g. for optics or transport properties (BoltzTraP), while using such a k-mesh for a full self-consistent calculation would not be necessary (and would be too expensive). In this case you want to do only one iteration (-i 1).

The procedure is the following:

- Do the calculation with the first k-mesh and "save" it when it is finished (do not execute clean_lapw since case.vectorhf should be present).
- mv case.klist_fbz case.klist_fbz_old
- mv case.klist_ibz case.klist_ibz_old
- mv case.outputkgenhf case.outputkgenhf_old
- mv case.klist_rfbz case.klist_rfbz_old (if the option -redklist is also used)
- mv case.klist_ribz case.klist_ribz_old (if the option -redklist is also used)
- Execute run.kgenhf_lapw to create the files for the new k-mesh.
- Run the HF calculation with -newklist:

  run(sp).lapw -hf -newklist (-i 1) ...

This option can be used simultaneously with -redklist, but not with -diaghf and -nonself. Note that when -newklist is used, the total energy (:ENE in case.scf) at the 1st iteration is wrong.

Band structure plotting

In order to make a plot of the band structure with hybrid functionals, it is more convenient to use the program run.bandplothf_lapw. After the self-consistent calculation is finished and saved (do not execute clean_lapw since case.vectorhf must be present), do the following steps:

- Create case.klist.band.
- Execute run.bandplothf_lapw with one or several of the following flags that were also used during the self-consistent calculation: -up/dn, -diaghf, -redklist, -model/mode2, -inlorig. Note that a parallel calculation of the band structure (with -p) can be done even if the scf calculation was not done in parallel (but you still need a file .machines) and vice-versa. You can also use -qtl to calculate the partial charges for band character plotting.
- Create case.insp.
- Execute x spaghetti with the switch -hf.

First, run.bandplothf_lapw calculates the semilocal orbitals (x lapw1 -band) for the k-points in case.klist.band. Then, the hybrid eigenvalues at these k-points are calculated (x hf -band). If -qtl is used, then the partial charges will be calculated (x lapw2 -band -qtl).

Density of states

The calculation of the DOS is the same as for the semilocal functionals, but using the additional flag -hf when executing lapw2 for the partial charges (x lapw2 -qtl -hf) and tetra for the DOS (x tetra -hf).
4.5. FLOW OF PROGRAMS

4.5.9 Slater, SmBJ and KLI potentials (“hf”-module)

The hf-module can also calculate the Slater [Slater, 1951], SmBJ [Becke and Johnson, 2006, Tran and Blaha, 2009] and KLI [Krieger et al., 1990] exchange potentials. The Slater potential (not to be confused with the LDA exchange potential of the homogeneous electron gas) is an average of the non-multiplicative HF potential. SmBJ is the mBJ potential (see Sec. 4.5.10) but using the Slater potential instead of the MGGA Becke-Roussel potential. The KLI potential consists of the Slater potential plus an additional term. These potentials are multiplicative, but nonlocal in the sense that they depend on the orbitals and require an integration at each point of space, therefore their calculations is rather expensive (same order of magnitude as for an HF calculation).

The steps to do a calculation with the Slater/SmBJ/KLI potential are the following:

► As for a standard LDA/GGA calculation, execute init_lapw to prepare the input files.
► Execute init_hf_lapw in order to prepare the input files for the hf-module.
► In case.in0, choose either the Slater (VX_SLATER), SmBJ (VX_SMBJ) or KLI (VX_KLI) exchange potential. If you do not want to add a correlation potential to the exchange one, then choose VC_NONE (this is mandatory with KLI). Since lapw0 -grr is not required you can delete case.in0.grr.
► If you have chosen SmBJ, then you need to follow the procedure explained in Sec. 4.5.10 to create the starting case.r2v and case.vresp files.
► In case.inhf, choose $\alpha = 1$ and do not use screening (replace T by F and remove the next line where $\lambda$ is specified).
► To start the calculation, do not execute run(sp).lapw, but the script run_vnonloc_lapw (see explanations below and Sec. 5.1.5) instead:

\[
\text{run\_vnonloc\_lapw (-sp) ...}
\]

About the input file case.inhf (see 7.6.2) for the Slater/SmBJ/KLI potentials:

► The Slater/SmBJ/KLI potential is multiplied by the parameter $\alpha$, whose default value is 0.25 (appropriate for hybrid functionals). Therefore, $\alpha$ should be set to 1 in order to have the full Slater/SmBJ/KLI potential.
► The value of nband is not used for the Slater/SmBJ/KLI potential. However, it is still necessary to replace “xx” by some integer.
► The meaning of gmax, lmaxe and lmaxv is the same as for hybrid functionals
► The tolerance tolu is not used for the Slater/SmBJ/KLI potential.

A few important points:

► The hf-module generates the file case.r2v_nonloc which contains the Slater/SmBJ/KLI potential multiplied by the electron density. This file is read by lapw0 in order to include the Slater/SmBJ/KLI potential into the total potential. If case.r2v_nonloc is not present for lapw0 (if this happen this should be only at the first iteration), then lapw0 will use the LDA exchange potential instead.
► The calculation of the Slater/SmBJ/KLI potential by the hf-module can be done in parallel. Both the k-point and MPI parallelizations can be used (simultaneously or only one of them). As usual, the k-point parallelization is over the k-points in the irreducible Brillouin zone. The MPI parallelization is implemented in the main subroutines for one of the loop of the double loops which run over the occupied bands. This means that the largest number of cores which makes sense to use is the number of k-points in the IBZ times the number of occupied bands.
► The scf convergence with the Slater/SmBJ/KLI potential can be extremely difficult to achieve. Therefore, it is more or less necessary to use the script run_vnonloc_lapw (instead of run(sp).lapw) to run the calculation. The script run_vnonloc_lapw runs the calculation with an inner/outer loops procedure similar to the one described in [Betzinger et al., 2010]
for the HF method. Within an inner scf loop, the Slater/SmBJ/KLI potential is kept fixed (frozen), which leads to rather fast convergence (the files are saved under a name which contains “fixed”). The outer loop consists of updating the Slater/SmBJ/KLI potential (the saved files are named with “updated”) right after an inner scf loop. Note that with KLI, the number of iterations for the outer loop can be huge (e.g., 100). For spin-polarized calculations, the flag “-sp” has to be used. See Sec. 5.1.5 for the available options and flages with run_vnonloc_lapw.

4.5.10 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 semilocal 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 a mBJ calculation (the purpose of the first five steps is just only to create the starting case.r2v and case.vresp files):

1. run a regular initialization and SCF calculation using LDA or PBE (it does not matter at all which functional you choose).
2. run init_mbj_lapw. This performs automatically the following steps:
   - create case.inm_vresp (cp $WIENROOT/SRC/templates/template.inm_vresp case.inm_vresp).
   - edit case.in0 and set ”R2V” option (instead of ”NR2V”) such that the XC potential is written in case.r2v.
3. run one more iteration (use run_lapw -NI -i 1) to generate the required case.r2v and case.vresp files.
4. run save the LDA (or PBE) calculation.
5. run init_mbj_lapw again. The second call (once case.inm_vresp is present) will do the following steps:
   - edit case.in0 and change the functional to option XC_MBJ (this is mBJ).
   - cp case.in0 case.in0_grr and choose EX_GRR VX_GRR in case.in0_grr. 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 procedure and lapw0 will be called twice, first with the input file case.in0_grr, then with case.in0.)
   - select a specific mBJ parametrization (see below) and creates the corresponding file case.in0abp.
6. Eventually, edit case.inm and choose the PRATT mixing scheme, although with recent mixer versions this should be hardly necessary.
7. run the mBJ SCF calculation.

In rare cases it could be that the default mixing scheme leads to convergence problems. The reason is that the mBJ potential also depends on the kinetic energy density which is not mixed in mixer. If such a convergence problem appears, you may have to restart the MSR1 mixing by removing the broyden files (rm *.broy) or even use the PRATT mixing. The PRATT mixing with the required small mixing parameter will be very slow, (otherwise leading to oscillations or even divergence) and thus later switch back to MSR1 after some initial (typical 5-20) scf-cycles.

The 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”. 
If you want to use other mBJ parameters than those defined in ([Tran and Blaha, 2009]), eg. the optimized values of ([Koller et al., 2012]) you can define them during init_mbj_lapw or directly in case.in0abp. Put 3 values A, B, e (default=-0.012, 1.023, 0.5), which determines the parameter c in mBJ according to eq. 7 or Table II. in ([Koller et al., 2012]).

4.5.11 DFT-D3 for dispersion energy

dftd3 calculates the dispersion energy and forces using the semi-empirical DFT-D3 method of [Grimme et al., 2010]. Since this method depends only on the positions of atoms (no dependence on the electron density) it is very fast and adds very little computer time. The dftd3 package is not included by default in WIEN2k, but can be downloaded from the website of the group of S. Grimme http://www.thch.uni-bonn.de/tc/index.php. When compilation is done, the executable dftd3 has to be copied in the $WIENROOT directory.

run(sp).lapw has to be executed with the -dftd3 switch:

run(sp).lapw -dftd3 ...

The user can either create the input file case.indftd3 (described in Sec. 7.2.2) by hand or let run(sp).lapw copy the default one from $WIENROOT/SRC/templates/.

The dftd3 package requires the file case.poscar (or case.xyz if periodic boundary conditions are switched off) created by the utility program struct2poscar, which is run automatically by run(sp).lapw.

The DFT-D3 method contains parameters which are specific to the exchange-correlation functional to which the dispersion energy is added. The functionals available in WIEN2k for which such parameters are available are the GGAs PBE, PBEsol, revPBE, RPBE, and BLYP, the MGGA TPSS, the GGA-hybrids PBE0, B3LYP, B3PW91 and HSE06 (corresponding to YS-PBE0 in WIEN2k) and the MGGA-hybrids TPSSh and TPSS0.

More detailed informations on the DFT-D3 method and available options are given in Sec. 7.2 and in the file man.pdf included in the dftd3 TAR file.

4.5.12 Nonlocal van der Waals functionals

The nlvdw module calculates the dispersion energy and potential with nonlocal van der Waals (NL-vdW) functionals [Dion et al., 2004] using the FFT-based method of Román-Pérez and Soler [Román-Pérez and Soler, 2009]. Details specific to WIEN2k can be found in [Tran et al., 2017].

The user has to copy the template input file case.innlvdw (described in Sec. 7.3.2) from $WIENROOT/SRC/templates/ and modify it if necessary.

run(sp).lapw has to be executed with the -nlvdw switch:

run(sp).lapw -nlvdw ...

A few important points:

▶ The default values in case.innlvdw for the plane-wave expansion cutoff $G_{\text{max}}$ (20 bohr$^{-1}$) and density cutoff $\rho_c$ (0.3 bohr$^{-3}$) should be rather well converged in most cases. However, for very weakly bound van der Waals systems, it may eventually be safer to check the convergence by repeating the calculation with a larger $\rho_c$ (e.g., 1 bohr$^{-1}$). Be aware that with such a larger value for $\rho_c$ it may be necessary to increase $G_{\text{max}}$ to a larger value (e.g., 30 bohr$^{-1}$). More details concerning the convergence with respect to $\rho_c$ and $G_{\text{max}}$ can be found in [Tran et al., 2017].
If the calculation of forces is not required (i.e., no optimization of atomic positions), then it is recommended to switch off the calculation of the NL-vdW potential in `case.innlvdw` in order to speed up the calculation. The NL-vdW potential affects only very little the density and electronic structure and is therefore essential only for the forces.

- `nlvdw` is parallelized with MPI which consists mainly of MPI FFTW (see Sec. 5.5).
- `nlvdw` needs the density on a grid in `case.xsf`. This file (and the necessary input `case.inxsf`) is prepared automatically by `prepare.xsf.lapw` called by the `run(sp) lapw` script.
- The script `run(sp) lapw` executes the module `lapw0` with the switch `-nlvdw (x lapw0 -nlvdw)` such that the NL-vdW potential (if required in `case.innlvdw`) is read from `case.r2v.nlvdw` and added to the semilocal potential.
- The NL-vdW energy is written in `case.scf` next to the label :ENLVDW and is added to the total energy by `mixer`.

About the kernel tables:

- The path of the kernel tables (specified in `x_lapw`) that are used for the calculations are `$WIENROOT/SRC_nlvdm/vdW_kernel_table` (for kernel type 1, generated with $Nr_points=2000, Nqs=30 and q_cut=10, which is more or less similar as in [Klimeš et al., 2011] and [Tran et al., 2017])

- `$WIENROOT/SRC_nlvdm/rVV10_kernel_table` (for kernel type 2, generated with $Nr_points=1024, Nqs=20 and q_cut=0.5, which is the default in other codes).

- The kernel table of type 1 which is the default in most other codes is `$WIENROOT/SRC_nlvdm/vdW_kernel_table_Nr_points1024_Nqs20_q_cut5` (generated with $Nr_points=1024, Nqs=20 and q_cut=5). If the user wants to use this kernel, then it has to be copied as follows:

  ```
  - cd $WIENROOT/SRC_nlvdm/
  - cp vdw_kernel_table_Nr_points1024_Nqs20_q_cut5 vdw_kernel_table
  ```

- Kernel tables with other parameters can be generated by using the stand-alone programs `generate_vdW_kernel_table.f` and `generate_rVV10_kernel_table.f` that are in `$WIENROOT/SRC_nlvdm/`. First, compile with `ifort -FR generate_vdW_kernel_table.f` or `gfortran -ffree-form generate_vdW_kernel_table.f` and then run the executable `a.out`.

The functionals:

The type of NL-vdW kernel and its parameter(s) are chosen in `case.innlvdw` (Sec. 7.3.2), however, as usual the semilocal part of the exchange-correlation functional has to be specified in `case.in0`. There is no restriction on the combination semilocal/NL-vdW. For well-known NL-vdW functionals, the keywords are shown below:

- **vdW-DF** [Dion et al., 2004]:
  - `case.in0`: EX_REVPBE EC_LDA VX_REVPBE VC_LDA
  - `case.innlvdw`: kernel type 1 and $Z_{ab} = -0.8491$

- **vdW-DF2** [Lee et al., 2010]:
  - `case.in0`: XC_GGA_X_RPW86 EC_LDA VC_LDA
  - `case.innlvdw`: kernel type 1 with $Z_{ab} = -1.887$

- **C09-vdW** [Cooper, 2010]:
  - `case.in0`: XC_GGA_X_C09X EC_LDA VC_LDA
  - `case.innlvdw`: kernel type 1 with $Z_{ab} = -0.8491$
4.5. FLOW OF PROGRAMS

- **optB88-vdW** [Klimeš et al., 2010]:
  - `case.in0`: EX_OPTB88 EC_LDA VX_OPTB88 VC_LDA
  - `case.innlvdw`: kernel type 1 with $Z_{ab} = -0.8491$

- **optPBE-vdW** [Klimeš et al., 2010]:
  - `case.in0`: EX_OPTPBE EC_LDA VX_OPTPBE VC_LDA
  - `case.innlvdw`: kernel type 1 with $Z_{ab} = -0.8491$

- **optB86b-vdW** [Klimeš et al., 2011]:
  - `case.in0`: EX_OPTB86B EC_LDA VXOPTB86B VC_LDA
  - `case.innlvdw`: kernel type 1 with $Z_{ab} = -0.8491$

- **rVV10** [Vydrov and Van Voorhis, 2010, Sabatini et al., 2013]:
  - `case.in0`: XC_GGA_X_RPW86 EC_PBE VC_PBE
  - `case.innlvdw`: kernel type 2 with $b = 6.3$ and $C = 0.0093$

- **vdW-DF-cx** [Berland and Hyldgaard, 2014]:
  - `case.in0`: XC_GGA_X_LV_RPW86 EC_LDA VC_LDA
  - `case.innlvdw`: kernel type 1 with $Z_{ab} = -0.8491$

- **rev-vdW-DF2** [Hamada, 2014]:
  - `case.in0`: XC_GGA_X_B86_R EC_LDA VC_LDA
  - `case.innlvdw`: kernel type 1 with $Z_{ab} = -1.887$
5 Shell scripts for running programs

Contents

5.1 Job control ........................................... 61
5.2 Utility scripts ........................................ 67
5.3 Structure optimization ............................... 74
5.4 Phonon calculations ................................. 79
5.5 Parallel Execution .................................... 80
5.6 NMR calculations: Chemical shift and Knight shift ......................................................... 87
5.7 Wannier functions (wien2wannier) .............. 92
5.8 Spontaneous Polarization, Piezoelectricity and Born Charges (BerryPI) ..................... 94
5.9 Getting on-line help .................................... 97
5.10 Interface scripts ...................................... 98

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

```bash
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

```bash
x -h or x_lapw -h
```

With some of the options the corresponding input files may be changed temporarily, but are set back to the original state upon completion.
CHAPTER 5. SHELL SCRIPTS

USAGE: x PROGRAMNAME [flags]

PURPOSE: runs WIEN executables: afminput, aim, arrows, broadening, cif2struct,
        clmaddsub, clmcopy, clminter, convham, conv2prim, dftd3, dipan, dmftproj,
        dstart, eosfit, eosfit6, filtvec, findbands, fleur2wien, hex2rhomb, hf,
        initxspec, irrep, joint, joinvec, kgen, kram, lapw0, lapw1, lapw2,
        lapw3, lapw5, lapw7, lapwdm, lapwso, lcore, lorentz, lstart, mini, mixer, nn,
        optimize, orb, pairhess, plane, rhomb_in5, sggroup, shifteig, spaghetti,
        struct2 cif, struct2poscar, struct_sfm_check, sumpara, supercell, symmetry,
        symmetso, telnes3, tetra, txspec, wannier90, w2w, w2waddsp, wplot, xspec

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
- ud -> runs up/dn-croossterm
- sc -> runs semicore calculation
- c -> complex calculation (no inversion symmetry present)
- p -> run lapw0/1/2/hf/so/dm/optic/dstart in parallel (needs .machines or .processes file)
- scratch dir/ -> defines (and makes) SCRATCH variable
  - grr -> lapw0 for mBJ or hf (using $file.in0_grr)
  - eece -> for hybrid-functionalss (lapw0, lapw2, mixer, orb, sumpara)
  - band -> for lapw1/2/3 bandstructures: uses list_band
  - orb -> runs lapw1 with LDA+U/OP or B-ext correction, mixer with dmat
  - it -> runs lapw1 with iterative diagonalization
  - noHinv0 -> runs lapw1 with iterative diag. without .Hinv
  - noHinvV0 -> runs lapw1 with iterative diag. writing new .Hinv
  - nohns -> runs lapw1 without HNS
  - nmat_only -> runs lapw1 and yields only the matrixsize
  - nmr -> runs lapw1 in NMR mode
  - innorig -> runs lapw2 but does not modify case.in1
  - emin X -> runs lapw2 with Emin=K (in $file.in2)
  - all K Y -> runs lapw2 with ALL and E-window K-Y (in $file.in2)
  - q1 -> runs lapw2 and calculates QTL
  - alm -> runs lapw2 and calculates ALM, BLM
  - qdmf -> runs lapw2 and calculates charges including DMFT (Aichhorn)
  - help_files -> runs lapw2 and creates helpXX files
  - vresp -> runs lapw2/mixer and creates case.vrespval/sum (TAU/meta-GGA)
  - fermi -> runs lapw2 with FERMI switch
  - efg -> runs lapw2 with EFG switch
  - hf -> runs lapw2 with Hartree-Fock/hybrid vectors
  - diaghf -> calculates only the diagonal elements of HF Hamiltonian
  - nonself -> calculates hf with RX only (no eigenvalues/vectors)
  - model23 -> calculates hf with a better HF scaling for memory
  - newklist -> HF/hybrid-DFT calculation starting from a different k-mesh
  - redklist -> HF/hybrid-DFT calculation with reduced k-mesh for the potential
  - slater -> calculation of the Slater potential by the HF module
  - gw -> write in case.gw the diagonal matrix elements of the HF/hybrid Hamiltonian for GW
  - fft -> runs kgen and generates a full mesh in the BZ
  - sf -> runs dstart only up to case.in0_std creation
  - super-> runs dstart and creates new_super.clmsum (and not $file.clmsum)
  - lecor-> runs dstart with $file.rspcloro (produces $file.clmsc)
  - sel -> use reduced vector file in lapw?
  - settol 0.00x -> run sgroppath with different tolerance
  - sigma-> runs lstart with case.inst_sgamma (autogenerated) for diff.dens.
  - dftd3-> run mixer with dftd3 forces
  - rxes -> run tetra using case.rxes weight file for RXES-spectroscopy.
  - rxeew E1 E2 -> run tetra and create case.rxes file for RXES for energies E1-E2
  - enefile -> spaghetti+tetra with case.energy instead case.q1 (only tot-DOS)
  - joinvec -> join case.energy_. files
  - select -> run arrows program with difference of two structures
  - copy -> runs pairhess and copies .minpair to .minrestart and .minhess
  - telnes -> run qtl after generating case.sfm based on case.inners
  - xspec -> run broadening with case.xspec instead of case.elnes
  - tx -> runs cif2struct using case.txt (see UD)
  - pp -> run wannier90 in "preprocessing mode"
  - wF N -> run wplot for Wannier function N
  - -fermi EF -> run findbands (unit:Ryd) / shift eig (unit:eV) with Fermi energy EF
  - -elex Y -> for findbands

USE: x -h PROGRAMNAME  for valid flags for a specific program
5.1. JOB CONTROL

Note: To make use of a scratch file system (usually a “local” file system for reducing the network or central fileserver load), you may specify such a filesystem in the environment variable `SCRATCH` (it may already have been set by your system administrator and must exist on all your nodes) or using the `--scratch` switch (directory will be created automatically if it does not exist). 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 Create the master input file `case.struct` (makestruct_lapw)

The primary input file for a case is called `case.struct`. It can be created by the `Struct Generator` of `w2web`, by some utilities like `cif2struct` or `xyz2struct` or using an interactive script `makestruct_lapw`. This script asks for lattice-type or spacegroup, atoms and their positions, and produces an intermediate file `datastruct`. The auxiliary programs `Tmaker` and `setrmt_lapw` converts this into `init.struct`, which must be copied to the proper location/filename by the user.

`makestruct_lapw` was provided by Morteza Jamal (m_jamal57@yahoo.com) and Peter Blaha.

5.1.3 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 quick-start 3 or the `makestruct_lapw` script 5.1.2), namely `case.struct` (see 4.3). (`case.inst` can be created automatically on the “fly” or using `instgen_lapw`, 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`.

```
init_lapw -h
```

Program: /zeus/WIEN2k/init_lapw
Purpose: initialisation of the l/apw-package WIEN2k
to be called within the case-directory
has to be located in WIEN-executable directory
needs case.struct file
Usage: init_lapw [OPTIONS] [FLAGS]
Flags:
  -h/-H ->help
  -b ->batch (non-interactive) mode (see possible options below, SGROUP is always ignored)
  -sp ->spin-polarized calculation
Options:
  -red X ->RMT reduction by X % (default: RMT not changed)
  -vxc X ->VXC option (default: 13 = PBE )
  -ecut X ->Energy seperation between core/valence (default: -6.0 Ry)
  -rkmax X ->RKMAX (default: 7.0, not changed)
  -lvns L ->LVNS_max (default: 4)
  -fermit X ->use TEMPS with smearing by X Ry (default: TETRA)
  -mix X ->set mixing to X (default: 0.2, not changed)
  -numk X ->use X k-points in full BZ (default: 1000)
  -s PROGRAM ->start with PROGRAM ($next)
  -e PROGRAM ->exit after PROGRAM ($stopafter)

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.

`init_lapw` may produce various WARNING or ERROR messages, when the default initialization might be unsafe. Ignoring ERRORS and in many cases also WARNINGS during the execution of this script, will most likely lead to errors at a later stage. Neglecting warnings about core-leakage creates `.lcore`, which directs the scf-cycle to perform a superposition of core densities, but usually you should rerun with a lower `-ecut` option (or your RMT spheres are not set properly).
CHAPTER 5. SHELL SCRIPTS

`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, RK-max, L-max for nonspherical elements (large spheres), k-mesh or mixing. Changes to `case.struct` by `nn` will be accepted, but by `sgroup` will be neglected. Please check the terminal output for ERRORS and WARNINGS !!!

### 5.1.4 Job control for iteration (run_lapw or runsp_lapw)

In order to perform a complete SCF calculation or even perform both, an optimization of internal atomic positions and a SCF calculation simultaneously, several types of scripts are provided with the distribution. For the specific flow of programs see chapter 4.5. For more information on atomic position optimization see chapter 5.3.2.

- 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

**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.)
- `-e PROGRAM` -> exit after PROGRAM ()
- `-l 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
- `-lnew 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
- `-scratch dir/` -> sets (and creates) scratch directory (for vector files)

**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 [speedname])
- `-it` -> use iterative diagonalizations
- `-it1` -> use iterative diag. with recreating H_inv (after basis change)
- `-it2` -> use iterative diag. with reinitialization (after basis change)
- `-nnoinv` -> 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
- `-inorig` -> if present, use case.inl_orig file; do not modify case.inl
- `-hf` -> HF/hybrid-DFT calculation
- `-diaghf` -> non-selfconsistent HF with diagonal HF only (only e_i)
5.1. JOB CONTROL

- model/2/3 -> nodea 2 and 3 calculate hf with a better MPI scaling for memory
- nonself -> non-selfconsistent HF/hybrid-DFT calculation (only \( E_\text{x}(\text{HF}) \))
- newklist -> HF/hybrid-DFT calculation starting from a different k-mesh
- slater -> calculation of the Slater potential by the HF module
- slater -> calculation of the KLI potential by the HF module
- dftd3 -> include the dispersion energy and forces with the DFT-D3 method
- min -> force optimization using MSR1a

CONTROL FILES:

- .lcore runs core density superposition producing case.clmsc
- .stop stop after SCF cycle
- .minstop in MSR1a mode (structure optimization) switches to MSR1
- .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 \( E_\text{x} \) analysis)

ENVIRONMENT VARIABLES:

- SCATCH 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
- -noorbdu -> use LDA+U without crossterms up-dn (needs also -so)

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 \( \mu \)Ry, 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 an-
other 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.

An additional switch -min will activate the optimization of the internal positions using the MSR1a
option in case.inn (see Sec. 5.3.2). Note, this option can take several hundreds of scf-cycles in
more complicated cases.

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 \( E_F - 0.2 \).
CHAPTER 5. SHELL SCRIPTS

The switch \texttt{-lnlorig} can be used to keep the present \texttt{case.inl} file unmodified (or to copy \texttt{case.inl.orig} back after \texttt{-lnlnew}).

The switch \texttt{-lnlnew N} preserves for \texttt{N} iterations the current \texttt{case.inl} file. After the first \texttt{N} iterations \texttt{write.inl.lapw} is called and a new \texttt{case.inl} file is generated, where the energy parameters are set according to the :EPLxx and :EPHxx values of the last scf iteration and the \texttt{-ql} value (see sections 4.4 and 7.5). 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 unsafe 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 \texttt{case.inl} file is saved in \texttt{case.inl.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. 7.5. It needs the \texttt{case.vector.old} file from the previous scf-iteration (and this file is created from \texttt{case.vector} when the -it switch is set) and an inverse of a previous Hamiltonian ($H^{-1}$) stored in \texttt{case.storeHinv}. When you change the Hamiltonian significantly (changing RKmax or local orbitals), reinitialize the iterative diagonalization either by “\texttt{touch .fulldiag}” (performs one full diagonalization) or “\texttt{touch .noHinv}” (recreates \texttt{case.storeHinv} files) or using the \texttt{-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 \texttt{N} iterations with the \texttt{-nohns N} switch.

The presence of the file \texttt{.lcore} directs the script to superpose the radial core densities using \texttt{dstart} and generating \texttt{case.clmsc}. It is created automatically during \texttt{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. \texttt{dstart} can also run in mpi-parallel mode, otherwise it can be slow for big cases.

The presence of the file \texttt{case.in0.grr} activates a second call of \texttt{lapw0}, which is necessary for modified Becke-Johnson potentials (see Section 4.5.10) or $E_{xc}$ analysis. It is also possible to calculate exact exchange (Hartree-Fock) and perform full hybrid-DFT calculations. However, such calculations are very expensive. They are activated using the \texttt{-hf} switch. More information can be found in Sec. 4.5.8

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

For magnetic systems which are difficult to converge you can use the script \texttt{runfsm.lapw \texttt{-m M}} (see section 4.5.3) for the execution of fixed-spin moment (FSM) calculations.

5.1.5 Job control for iteration with the Slater/SmBJ/KLI potentials (run.vnonloc_lapw)

In order to achieve scf convergence with the Slater/SmBJ/KLI potentials (see sec. 4.5.9) it is necessary to use the script \texttt{run.vnonloc_lapw}. The script \texttt{run.vnonloc_lapw} runs the calculation with an inner/outer loops procedure similar to the one described in [Betzinger et al., 2010] for the HF method. Within an inner scf loop, the Slater/SmBJ/KLI potential is kept fixed (frozen), which leads to rather fast (and very cheap) convergence (the files are saved under a name which contains “fixed”). The outer loop consists of updating the Slater/SmBJ/KLI potential (the saved files are named with “updated”) right after an inner scf loop. For spin-polarized calculations, the flag “-sp” has to be used. Many of the options and flags of \texttt{run(sp).lapw} (-ec, -cc, etc.) can be used with \texttt{run.vnonloc_lapw}. All options and flags can be listed with the \texttt{-h} flag:
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
```

can save the calculation by saving the following files (as well as all inputs)

- `case.clmsum`
- `case.scf`
- `case.dmat`
- `case.vorb`
- `case.struct`

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. Except when using the `-o` switch, 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
- `-o`  old scheme, does not save input files
- `-f`  force save_lapw to overwrite previous saves of the same name
- `-d` directory  save calculation in directory specified
- `-s`  silent operation (no output on screen)
- `-band`  saves case.output1/so, qtl, irrep and spaghetti files
- `-dos`  saves case.int, qtl and dos files
- `-eels`  saves elnes, innes, broadspec and qtl files
- `-optic`  saves case.symmat,joint,epsilon,sigma,eloss,absorp,klist,kgen,inop,injoint,inkram ... files
- `-xspect` saves xspec, corewfx, m1/m2, inxs, qtl and int files

Note: for DOS, bandstructure, xspec, eels or optic, there is no corresponding `restore_lapw` option, but files must be handled by hand.
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 Reduce atomic spheres and interpolate density (reduce_rmt_lapw)

```
reduce_rmt_lapw [ -r XX -up]
```

When a structure optimization (MSR1a and `run(sp)_lapw` or `min_lapw`) fails because of overlapping spheres, this script will reduce the spheres (default: 3 % or use `-r XX`) and interpolate the density inside the spheres to the new radial mesh. Setting the switch `-up` will do it for `clmsum`, `clmup` and `clmdn` files.

5.2.4 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.

Options to clean_lapw are:
- **-h** help
- **-s** silent operation (no output)
- **-r** recursively clean all directories starting from the current one

5.2.5 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
5.2. UTILITY SCRIPTS

-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)

5.2.6 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

5.2.7 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 in the STRUCTEDITOR@w2web or explicitly using:

setrmt_lapw case [-r X ] [-a XX:A,YY:B,... ] [-orig]

where case gives the head of the case.struct file. You may specify a reduction (-r) of the RMTs by X percent in order to allow for structural optimizations. If you already know which RMT values you want to know for a certain element, you can fix them using eg. (-a Mg:1.9). The new setrmt_lapw version knows optimal RKmax values for all atoms and makes a finer tuning of the different RMTs. with (-orig) you can go back to the old scheme which distinguishes only between H, sp- and d-elements. It creates case.struct_setrmt with the modified RMTs.

5.2.8 create_add_atom_clmsum_lapw

The script create_add_atom_clmsum_lapw creates a better starting density for a case, where you already have a scf-solution for a “similar” case. “Similar” means, that the new and old case are identical except for ONE atom (adding an adsorbate,...).

This script is “experimental” and only for experienced users. It is usefull in BIG cases, which are difficult to converge from init_lapw. Modifications and adaptions to a specific case are probably necessary.
5.2.9 Create case.int file (for DOS) (configure_int_lapw)

This script creates the input file case.int for the program tetra and allows to specify 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

5.2.10 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).

5.2.11 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 useful for killing “k-point parallel” jobs.

5.2.12 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 Å, e/Å³,... and produces critical_points_ang.

It is used with: extractaim_lapw case.outputaim
5.2. UTILITY SCRIPTS

5.2.13 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 GNUPLPLOT. This plot is updated in regular intervals.

You can call scfmonitor_lapw using:

```
```

-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 GNUPLPLOT 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.

5.2.14 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 independent: 
atom iii dependent: 
vector quantities: 
where magnitude z z is the default

For vector quantities like :FGLiii or :POSiii (useful with case.scf.mini) one can specify the respective coordinate by adding x/y/z to the corresponding labels.
5.2.15 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).

5.2.16 Check parallel execution of lapw1 (testpara1_lapw)

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

5.2.17 Check parallel execution of lapw2 (testpara2_lapw)

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

5.2.18 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.

5.2.19 initso_lapw

initso_lapw helps you to initialize the calculations for spin-orbit coupling. It helps together with make_inso_lapw (based on an idea of 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 7.7. Since forces are not correct for atoms with SO, it can be very useful to suppress SO for light atoms (eg. the O-atoms in UO$_2$), because then one can optimize the O-positions.

5.2.20 init_hf_lapw

init_hf_lapw helps you to initialize the calculations for hybrid-DFT functionals. It creates several files (case.inhf, case.in0.grr), selects YS-PBE0 (see [Tran and Blaha, 2011]), changes some input files (case.in0) and calls run.kgenhf_lapw to generate the k-mesh for the HF calculation. It takes -up for spin-polarized cases.

For details of hybrid-DFT calculations see 4.5.8.
5.2. UTILITY SCRIPTS

5.2.21 init_mbj_lapw

init_mbj_lapw helps you to initialize the calculations for a TB-mBJ (see [Tran and Blaha, 2009]) calculation, which usually requires a couple of steps done in proper order.

A proper sequence would be:

- `init_mbj_lapw`
- `run_lapw -i 1`
- `init_mbj_lapw`
- `save_lapw xxxx.pbe`
- `run_lapw`

The first call to `init_mbj_lapw` creates `case.inm_vresp` and sets “R2V” in `case.in0`. The second call of `init_mbj_lapw` creates `case.in0` and `case.in0_grr` with the proper input for mBJ. It also lets you select parameters of the original TB-mBJ potential, or the later adaption to semiconductors or insulators, or the original BJ method.

For details of TB-mBJ calculations see 4.5.10.

5.2.22 vec2old_lapw

vec2old_lapw moves `case.vector` files to `case.vector.old`. Usually called automatically just before `lapw` 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 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 activated using the `-vec2pratt` switch in `run_lapw`.

5.2.23 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`) from the old positions and adds an atomic superposition density from 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.
5.2.24 makescratch

makescratch scratch-dir-name checks the existence of the directory and eventually creates it (up to 3 levels deep). Usually called automatically by other Wien2k-scripts.

5.3 Structure optimization

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

Package optimize

The auxiliary 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 (tetrahedral and hexagonal lattices)
3. VARY C/A RATIO with CONSTANT VOLUME and B/A (orthorhombic lattice)
4. VARY B/A RATIO with CONSTANT VOLUME and C/A (orthorhombic lattice)
5. VARY A and C (2D-case) (tetrahedral 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) (tetrahedral and hexagonal lattices)

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

```bash
#!/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}
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 equilibrium 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 or gibbs_lapw to find the minimum total energy and the equilibrium volume (or c/a or b/a). Supported equation of states include the EOS2, Murnaghan and Birch-Murnaghan EOS.

\[
\text{grepline} :\text{ENE} \text{ '*.scf'} 1 > \text{case.analysis} \\
\text{grepline} :\text{VOL} \text{ '*.scf'} 1 >> \text{case.analysis}
\]

Alternatively you can also use eplot_lapw directly and the case.analysis file is generated automatically:

\[
\text{eplot_lapw -a vol}
\]

or

\[
\text{eplot -a "*" will analyse all scf files "vol*.scf"} \\
\text{eplot -a pbe will analyse all scf files "vol*pbe.scf"}
\]

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 8.5 and IRelast in sec 8.8.

Package 2DRoptimize

This program was contributed by:

Morteza Jamal
Ghods City-Tehran,Iran
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 (see also [Reshak and Jamal, 2013]) performs a convenient 2D structure optimization (Volume and c/a for tetragonal, rhombohedral or hexagonal spacegroups). After initialization of a case, one generates a set of structures and a job-file 2Doptimize.job using the command

\[
\text{set2D_lapw}
\]

This calls setup2D and you have to specify 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

\[
\text{ana2D_lapw}
\]
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. ana2D_lapw checks the sensitivity of the results with the order of fitting (3, 4 or 5th order polynomials) and lets you select the best one. Note: Fits of high order (and few “data points”) may lead to artificial results due to unphysical oscillations 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.

At the end, ana2D_lapw calculates a and c lattice constants (and $a_R$, $\alpha_R$ for rhombohedral compounds) and checks the sensitivity of them to the order of fit (order of fit=3 or 4 or 5) when it finds the equation of c/a vs. volume and stores in fitorder.

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 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 answsocf=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 -min where in case.inm the switch MSR1 will be modified to MSR1a 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 $.gt. 5 \text{mRy/bohr}$), 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:
5.3. STRUCTURE OPTIMIZATION

▶ 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”. save_lapw xxx the original calculation and 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 $WIEN-ROOT/SRC_mixer/README.5.2.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, 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 (e.g. 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
5.4. PHONON CALCULATIONS

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 oscillations 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.tmpM 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.d4s must be transferred to UNIX. Here you would run WIEN2k-scf calculations for all displacements and collect the resulting forces, which will be transferred 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.d4s 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
CHAPTER 5. SHELL SCRIPTS

- 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 file 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 Linux 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 very large network bandwidth.

The other parallelization method is based on fine grained methods, MPI, ScaLAPACK, and (optionally) ELPA. 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 (Infiniband) or a shared memory machine. Although for small systems (less than 50 atoms/cell) it is 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 larger problems and most parts of WIEN2k. 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 $\frac{NMAT}{\sqrt{\text{processors}}} \geq 1500$ should hold.

The k-point parallelization can use a dynamic load balancing scheme and is therefore also usable 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 first if you have more than 1 k-point).
5.5. PARALLEL EXECUTION

5.5.1 k-Point Parallelization

Parts of the code are executed in k-parallel, namely lapw1, lapwso, hf, lapw2, lapwdm and optic, qtl, irrep, nmr. 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 a cluster in such a way, that on all machines the path names are the same.

Remote login: ssh to all machines must be possible without specifying a password. This will be handled automatically for a single shared memory machine and when you have specified “shared memory” during site_config (setenv USE_REMOTE 0 in $WIENROOT/parallel_options). Otherwise you must 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 site_config.lapw is executed (see chapter 11).

5.5.2 MPI parallelization

Fine grained MPI parallel versions are available for the programs dstart, lapw0, lapw1, lapwso, hf, nmr, nlvdw and lapw2. This parallelization method is based on parallelization libraries, including MPI, ScaLapack, PBlas, ELPA, and FFTW_2 or FFTW_3 (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.

The parallelization affects the naming scheme of the executable programs: the fine grained parallel versions of lapw0/1/2/so, dstart, hf, nmr, and nlvdw are called lapw0_mpi, lapw1_mpi, lapw2_mpi, lapwso_mpi, dstart_mpi, hf_mpi, nmr_mpi, and nlvdw_mpi. These programs are executed by calls to the local execution environments, as in the sequential case, by the scripts x, dstartpara, lapw0para, lapw1para, lapwsopara, hfpara, nlvdwpara and lapw2para. On most computers this is done by calling mpirun and this must 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.

---

CHAPTER 5. SHELL SCRIPTS

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`, `hfpara`, `nlvdwpara` `lapw2para` (when using `-p`), and parallel programs `dstart_mpi`, `lapw0_mpi`, `lapw1_mpi`, `lapw2_mpi`, `hf_mpi`, `nlvdw_mpi`, and `lapw3_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
3:gamma:2 delta epsilon
3:delta:4 epsilon:4
residue:delta:4
balance:
lapw0:gamma:2 delta:2 epsilon:4
dstart:gamma:2 delta:2 epsilon:4
nlvdw: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). Because nowadays one works mostly on homogeneous clusters, the weight (speed) of a machine is ONLY honored when the keyword `balance:` is also given. 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 computers `delta` and `epsilon` cooperate in a fine grained parallelization on the solution of 3 k-points, and four processors of computers `delta` and `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` (and some others) to determine which `case.vector*` to read. In case you need to create a `.processes` file for a NEW `.machines` file and don’t want to run `lapw1` (for instance in a PBS-job with “x lapw1 -p -qtl”) you can issue: `x lapw1 -p -d [-up]` to create an updated version of this file.

A “granularity” different from 1 (use eg. 3) allows for some load balancing in heterogeneous environments. Suppose you have 10 k-points and 2 nodes, granularity:1 will start 2 jobs with 5 k-points each. However, if node 1 is heavily overloaded, node 2 will idle for quite some time and time will be wasted. With a larger granularity we would decompose the load into 4 or 6 parts. Two jobs would start first, but the next parts go to the node which is free because it has finished earlier. 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 (less file I/O).

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 is insured since the WIEN2k,16 version unless the “balance:” keyword is set. By default it will now distribute all k-points at once. If you have 9 k-points and 2 cores, you get 2 jobs with 5 and 4 k-points, respectively.

The lines

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

defines the computers used for running `lapw0_mpi`, `dstart_mpi`, and `nlvdw_mpi`. In this example the 8 processors of the computers `gamma`, `delta`, and `epsilon` run `lapw0_mpi`, `dstart_mpi`, and `nlvdw_mpi` in parallel. The parallel `dstart` is useful for big cases, where core-leakages occurred and a core-density superposition is done automatically (activated by the file `.lcore`) during scf. Please note, parallelization in `lapw0` and `dstart` is done mainly over atoms, thus the number of useful cores is in general different than for `lapw1/2/so/hf`. The parallelization in `nlvdw` concerns mainly the FFT.

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 SLURM, 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_user/faq](http://www.wien2k.at/reg_user/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 is split into subsets according to the weights specified in the .machines file:

\[
newweight_i = \left\lfloor \frac{weight_i \times klist}{granularity \times \sum_j weight_j} \right\rfloor
\]

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 testparal_lapw (see 5.2.16) and testpara2_lapw (see 5.2.17) 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](image-url)
5.5.7 On the fine grained parallelization

The following parallel programs use different parallelization strategies:

- **dstart_mpi** is parallelized over the atoms and the K-vectors. 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 *dstart_mpi* is quite small. It uses an extra line “dstart:” in *machines* to specify the parallelization.

- **lapw0_mpi** is parallelized over the number of atoms and with a parallel FFT, which is important in case you have large FFT grids. 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. It uses an extra line “lapw0:” in *machines* to specify the parallelization.

- **lapw1_mpi** uses a two-dimensional processor setup to distribute the Hamilton and overlap matrices. For higher numbers of processors two-dimensional communication patterns (4x4=16, 8x8=64,...) are clearly preferable to one-dimensional communication patterns (never use 47 cores, as it gives a 47x1 pattern).

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 WIEN2k the processor grid $P \times Q$ is chosen such that the shape of the grid is “as square as possible”, i.e. the sum of $P$ (the larger grid dimension) and $Q$ (the smaller grid dimension) is minimal. This is done heuristically during the setup of the parallel calculation. Square grids ($P \times P$, i.e. 4, 9, 16, ... processors), if possible, give best performance due to ScaLAPACK. However, other grids are also possible (e.g. 4x2=8, 8x4=32, ...). The default setup for rectan-
gular grids is $P \times Q$, but for certain cases (depending on problem size, number of processors, ...) a $Q \times P$ grid can perform better – this can be set in the lapw1 input file (see also Sect. 7.5.3).

k-point and mpi-parallelization can be used at the same time and are specified by the lines “speed:hostname:number_of_cores” in .machines.

**hf** If you use the `-hf` option for run_lapw (“full hybrid” scheme, see Sect. 4.5.8, or the Slater/SmBJ/KLI potential, see Sect. 4.5.9), then the `hf` program will be by far the most time consuming part. For the hybrid functionals, the MPI-parallelization is done over the number of occupied bands or the matrix elements of the second variational Hamiltonian. There are two modes of MPI-parallelization (`-model` or `-mode2`, see details in Sect. 4.5.8). Choose first the best k-parallelization (if you have more than one k-point) and then a MPI-parallelization for the loops over the occupied bands/matrix elements. In the case of the Slater/SmBJ/KLI potential, this is one of the loop of the double loops over the occupied bands that is parallelized with MPI. The parallelization follows that of lapw1 as specified in .machines, although the script uses .processes, which is created at the lapw1 step.

**nlvdw** consists of a parallelization for the FFT (MPI FFTW), which is important in case of large FFT grids. This method leads to good scalability in particular for the memory.

**lapwso** is parallelized over the Hamiltonian. The size is determined by $NE^2$, where $NE$ is the number of eigenvalues in lapw1 (determined by EMAX in *case.in1*). Since this size is usually much smaller than the Hamiltonian of lapw1, try to use quadratic processor grids (4x4=16, 8x8=64). Memory size is larger than for the sequential code, but scales with $\sqrt{N}$. The parallelization follows that of lapw1 as specified in .machines, although the script uses .processes, which has been created in the lapw1 step.

**lapw2** 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 (usually 2-8). This is in principle not so efficient, but must be used if the memory requirement is too big (typically when lapw2 mpi crashes “without” reasons) or the network is slow and using more cpu-time but less network traffic is more efficient. Test it out for your hardware and specific case). You set it in .machines using

```
lapw2_vector_split:2
```

Otherwise, the parallelization follows that of lapw1 as specified in .machines, although the script uses .processes, which is created at the lapw1 step.

**nmr** in mode “current” supports the same parallelization strategy (mixed k-point and mpi-scheme) as lapw1 or lapw2. The keyword

```
nmr_integ: node1 node2 ...
```

allows for an additional mpi-parallelization (over the atoms) in mode “integ” (see description in chapter 5.6).

If more than one k-point is distributed at once to lapw1 mpi or lapw2 mpi, they 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 or 1 Gbit/s Ethernet network is not recommended, even for large problem sizes.
5.6 NMR calculations: Chemical shift and Knight shift

5.6.1 Chemical shift

Introduction

The calculation of the magnetic shielding tensor $\sigma$ in insulators is based on a linear response theory described in [Laskowski and Blaha, 2012a, Laskowski and Blaha, 2012b, Laskowski and Blaha, 2014]. In short, the calculation of the NMR shielding tensor requires eigenvectors computed at seven different k-meshes: original and shifted by $q$ in $+/-x,y,z$ direction, where $q$ is small compared to the BZ size. Those eigenvectors are then used to compute the induced current and magnetic susceptibility. The induced current is afterwards integrated (Biot-Savart) to get the NMR shielding tensor.

The script `x_nmr_lapw` helps you to perform all the necessary steps and together with the NMR-program (see chapter 8.16) allows you to calculate the chemical shielding (and further the chemical shift with respect to some reference compound). It requires a converged scf-calculation of your case (and for the time being, the system should be insulating, see below for Knight shifts).

The implemented method uses an enriched APW basis set (extended number of local orbitals, called NMR-LOs). The setup of NMR-LOs is communicated to other programs (for instance lapw1) via the file `case.in1_nmr` file (`case.in1c` for cases without inversion symmetry). Therefore after converging SCF or restoring a previously saved calculation, one has to create `case.in1_nmr`. The `case.in1_nmr` file should be generated using:

```
x_nmr_lapw -mode in1 [parameters]
```

The important parameter here is “-nodes val”, where val is an integer used to determine the number of NMR-LOs in each orbital quantum number $l$ (see [Laskowski and Blaha, 2012a, Laskowski and Blaha, 2014] for details). The default value (8) gives well a converged tensor, but it may also lead to an unnecessarily large basis size. In such cases the number of NMR-LOs may be reduced using a smaller number “val” (eg. 5), or by using “-focus atom” option that decreases the number of NMR-LOs for atoms other than the one specified. By default the algorithm implemented here adds NMR-LOs to the bases for all orbital numbers up to $l+1$, where $l$ is the maximal explicitly specified orbital in case.in1. In a case where the magnetic susceptibility needs to be computed precisely, an $l+2$ limit may be necessary to reach full convergence. In such cases it is required to add an extra entrance for the next $l$-value in case.in1 with a default 0.3 linearization energy (eg. a $l=2$ line for an O atom).

You may also consider to run `x_kgen` and create a (finer) k-mesh for the NMR calculation (in any case, the k-point dependency of the NMR tensor should always be tested explicitly by at least 2 different k-meshes).

After successful generation of a fine k-mesh and the `case.in1_nmr` file the NMR shielding tensor can be computed using:

```
x_nmr_lapw [parameters]
```

By default the `x_nmr` script will execute sequentially the following steps (you don’t need to call them explicitly):

1. shifted k-mesh generation based on the existing k-mesh generated previously
   ```
x_nmr_lapw -mode klist
   ```
2. eigenvectors (lapw1)
In a case where spin-orbit coupling needs to be included ("x.nmr -so") the proper eigenvectors vectors are generated with:

```
x.nmr_lapw -mode lapw1
```

Similarly, for hybrid-DFT calculations the eigenvectors will be computed by

```
x.nmr_lapw -mode hf
```

The eigenvectors are computed sequentially in subdirectories:

- `nmr_q0` (original k-mesh)
- `nmr_pqx` (shifted in (+q,0,0) in Cartesian frame)
- `nmr_mqx` (shifted in (-q,0,0) in Cartesian frame)
- `nmr_pqy` (shifted in (0,+q,0) in Cartesian frame)
- `nmr_mqy` (shifted in (0,-q,0) in Cartesian frame)
- `nmr_pqz` (shifted in (0,0,+q) in Cartesian frame)
- `nmr_mqz` (shifted in (0,0,-q) in Cartesian frame)

If you are using a SCRATCH variable different from `''./''`, it is recommended to define a unique scratch directory with

```
x.nmr_lapw -scratch /scratch/caseA
```

in order to avoid collisions between multiple NMR calculations running simultaneously.

3. weight files

```
x.nmr_lapw -mode lapw2
```

4. core wave functions

```
x.nmr_lapw -mode lcore
```

5. induced current density and magnetic susceptibility

```
x.nmr_lapw -mode current
```

The current is written to `case.current.sp(x,y,z)`, `case.current.int.(x,y,z)`, where x,y,z are the Cartesian directions of external magnetic field. The current density is UNSYMMETRIZED with respect to irreducible BZ. In order to get a symmetrized current for plotting purposes the full BZ sampling has to be used (`x kgen -fbz`). The magnetic susceptibility is written to `case.xim`.

6. integration of current density

```
x.nmr_lapw -mode integ
```

The full NMR tensor and other related quantities can be found in `case.outputnmr.integ`. The isotropic chemical shift $\sigma_{iso}$ and its anisotropy is printed under the label "NMR-TOTxxx" (in ppm) and :NMRASYxxx (Haeberlen convention):

```
>NMRTTOT001 ATOM: Te 1 NMR(total/ppm) Sigma-ISO= 1295.27 Sigma_xx= 1356.01 Sigma_yy= 1356.01 Sigma_zz= 1173.79
>NMRASY001 ATOM: Te 1 NMR(total/ppm) ANISO(delta-sigma)= -182.21 ASYM(eta) = 0.000 SPAN= 182.21 SKEW=-1.000
```

The steps 1) to 6) are executed one after another by `x.nmr` script, there is no need to run through them manually. However if there is need to recompute the current without changing eigenvectors (for analysis purposes), steps 5) and 6) can be executed using

```
x.nmr_lapw -noinit
```

Or when one needs to compute only initialization steps 1) to 4)

```
x.nmr_lapw -initonly
```

may be used.
5.6. NMR CALCULATIONS: CHEMICAL SHIFT AND KNIGHT SHIFT

Options

All options of the x_nmr script can be seen using:

    x_nmr_lapw -h

-h/h          print this message
-mode modeid  runs in specific mode given by modeid. If mode is not defined,
              runs sequence needed for actual calculations (klist,lapw1,[lapwso],
              lapw2, lcore, current, integ), however preceding execution in mode inl
              is still required. modeid can be:
              in1 (initialize case.in1_nmr, adds extra LO)
              testval (testing case.in1_nmr)
              klist (initialize shifted k-lists )
              lapw1 (executes lapw1)
              lapwso (executes lapwso, only after lapw1 step)
              hf (runs hf on top of lapw1, only after lapw1 step)
              lapw2 (executes lapw2 for weights, only after lapw1, lapwso or hf)
              lcore (executes lcore)
              current (generate induced current)
              integ (integrates current and computes nmr shielding parameters)
              plot (plot of the induced current, uses extra input file
                   case.innmrplot, generated automatically if not present)

-noinit       executes mode current and integ
-initonly     executes modes klist, lapw1, [lapwso], lapw2, lcore
-so           executes mode lapwso
-orb          adds LDA+U to lapw1 or lapwso
-hf           executes mode hf between lapw1 and lapw2 (as this takes long time,
              you certainly should run this in parallel. If you have more cores,
              use -hf -hfdir [nmr_q0, nmr_pqx, nmr_mqx ....] in parallel)
-hfdir subdir  prepares HF vectors (starting from lapw1 and ending with lcore)
               for the subdir=[q0, pqx, mqx, ....]. It allows more parallelization
               as all *subdir*s can be run with a different .machines file in parallel.
-redklist     uses a reduced k-list (case.klist_rfbz) for the HF potential
              (note, the general HF k-mesh must be the same as in the scf)
-newklist     uses the option newklist for the HF potential
-diaghf       diagonal approximation to HF (only eigenvalues updated)
-p            run in k-point or mpi parallel mode
-quota XXX    calculations in junks of XXX k-points (save disk space)
-case name    set the casename to name, otherwise the current dir name is used
-up           include spin polarisation (up spin)
-dn           include spin polarisation (dn spin)
-save dir     saves result in directory dir
-scratch scratch_dir sets (and creates if necessary) the scratch directory for
                    storing vectors

Mode specific parameters (ignored by others):

mode: in1
    -nodes val      number of nodes of the top radial function, default = 8
    -focus val      index or name of an atom of interest, if not set then all
                    maximum allowed overlap between top (energy) radial function
                    from in1 and NMR LO (default 0.6)
mode: testval
    -up/dn          include spin polarization (up/dn spin)
    -orb            add LDA+U switch to lapw1
mode: klist
    -q val          sets the q to value, if not defined uses default of 0.005
mode: lapw1 / lapw2 / lapwso
    -p              run in k-point or mpi parallel mode
-up/dn include spin polarization (up/dn spin)
-orb add LDA+U switch to lapw

mode: hf
-up/dn include spin polarization (up/dn spin)
-hfdir subdir prepares HF vectors (starting from lapw1 up to lcore) for the subdir=[q0, pqx, mqx, ....]. It allows additional parallelization as all "subdir"s can be run with a different .machines file in parallel.
-redklist allows to use a reduced k-list (case.klist_refbz) for the HF potential (note, the general HF k-mesh must be the same as in the scf)
-newklist uses the option newklist for the HF potential
-diaghf diagonal approximation to HF (only eigenvalues updated)

mode: current
-up/dn include spin polarization (up/dn spin)
-so use lapwso vectors
-hf use hf vectors
-emin val overrides the valence bands minimum
emax val overrides the valence bands maximum
-lemin val sets lowest valence band to val
-lemax val sets highest valence band to val
-filt_cxyz_o iat 1 filter coupling matrix element (<OS|COUPOP|ES>, make_cxyz) the occupied states <OS|. Leaves only nonzero alm for iat and l. 
-filt_cxyz_q iat 1 filter in coupling matrix elements (<OS|COUPOP|ES>, make_cxyz) the empty states |ES>. Leaves only nonzero alm for iat and l.
-filt_curr_o iat 1 filter in current density (make_current_sp,j(r)=<OS|JOP|FOP>) the occupied states OS. Leaves only nonzero alm for iat and l.
-filt_curr_fop iat 1 filter in current density (make_current_sp,j(r)=<OS|JOP|FOP>) the perturbation w-f |FOP>. Leaves only nonzero alm for iat and l.

For all -filt_* if (iat .eq. 0) do only interstitial contribution
For all -filt_* if (l .lt. 0) apply and sum all l channels
-nocc do not add core states to the Green function
-noduc do not add du (radial derivative of u) to the Green function
-scissor val applies scissor shift to conduction bands
-coreonly only core contribution
-xionly calculate only macroscopic magnetic susceptibility
-noxi do not calculate macroscopic magnetic susceptibility
-fbz k-sampling uses full BZ (no symmetrization of xi)
-metal should be set in case of metals
-kbT XX sets kbT for Fermi level smearing in metals for Green function

mode: integ
-nocore subtract core contribution

mode: plot (note: current is not symmetrized, must use full BZ sampling)
-nocore subtract core contribution
-up/dn include spin polarization (up/dn spin)

Additional notes

Parallelization:

```
x_nmr_lapw -p
```

will execute lapw1, lapw2 and x_nmr -mode current -p in k-point parallel mode following the standard WIEN2k scheme. The standard .machines file is used in this case. A mixed k-point/mpi parallelization (if more then one core is assigned to one k-point) is also implemented for x_nmr -mode current -p. The integration step x_nmr -mode integ -p supports mpi parallelization over atoms. In order to use it, the following line has to be added to the .machines file:

```
nmr_integ: $proc_list
```

where $proc_list is a list of processors.
It is possible to combine hybrid-DFT and nmr calculations, but note that this is quite expensive, in particular because of additional NMR-local orbital AND the need for ALL eigenvalues, which makes the \textit{hf} step MUCH more expensive than for a normal scf calculation (and we need calculations for 7 different k-meshes). We therefore recommend a good parallelization (if possible, over ALL k-points and in addition with mpi over the (un)occupied bands). After mode in1 run:
\begin{verbatim}
x_nmr_lapw -p -hf
\end{verbatim}
or, if you have enough cores create several different \texttt{.machines} files and run the 7 directories in parallel:
\begin{verbatim}
cp .machine_q0 .machines
x_nmr_lapw -p -hfdir q0 &
cp .machine_pqx .machines
x_nmr_lapw -p -hfdir pqx &
\end{verbatim}

\textbf{Analyses :}
\begin{verbatim}
x_nmr_lapw -p -noinit -emin xx [-emax yy]
\end{verbatim}
allows you to separate the contributions to the magnetic shielding according to the energy range (in Ry) of the valence bands (eg. the contributions from a "p-band" and a "d-band", ...). The switch \textit{-noinit} runs only the modes current and integ. Additional analysis is possible with the \textit{-filt} options, but requires some understanding of the underlying formalism (see the NMR papers by [Laskowski and Blaha, 2012a, Laskowski and Blaha, 2012b, Laskowski and Blaha, 2014, Laskowski and Blaha, 2015a]).

\textbf{Current plotting :}
You can also plot the induced current (it needs the \texttt{dx} Dataexplorer software), but since the current is not symmetrized, you need to run first with a full k-mesh. Use
\begin{verbatim}
x kgen -fbz # for plotting purposes this can be on a smaller k-mesh
x_nmr_lapw
x_nmr_lapw -plot # prepares current.dx and current_x/y/z.dx files
current2dx_lapw
\end{verbatim}

\section*{5.6.2 Knight shifts}
For paramagnetic metals, the \textit{Knight shift} dominates usually the \textit{Chemical shift}, which comes from the Fermi-contact term due to the spin-polarization at $E_F$[Laskowski and Blaha, 2015b, Laskowski et al., 2017]. The contact term can be calculated in an extra directory using a spin-polarized setup and a magnetic field applied self-consistently in \texttt{lapw0}. Copy \texttt{case.struct} into a new directory and execute:
\begin{verbatim}
instgen -nm # generate nonmagnetic atomic configurations
init_lapw -sp -fermit 0.004 -numk XXX ... # use a very good k-mesh and fermi-smearing
runsp.c_lapw -c 0.00001 [-p] ... # run scf with zero moment
cp $WIENROOT/SRC/templates/case.vorbup100T case.vorbup # contains the 100 T field
cp $WIENROOT/SRC/templates/case.vorbdn100T case.vorbdn
runsp_lapw -orbc -cc 0.000001 [-p] ... # scf calculation with field
grepline :HFF0XX case.scf # get the hyperfine field for atom XX
save_lapw 100T.XXX-kpoints.fermit-4 # save the calculation
\end{verbatim}
Now create a better k-mesh (a rough estimate is to use about 500000 k-points for a one-atomic unit cell, and this mesh can be reduced by division with the number of atoms/cell) and repeat the runsp_lapw step. Check convergence. Then modify case.in2(c) and choose another temperature smearing (eg. 2mRy and/or 6 mRy). The contact term of the shielding can be obtained from the HFF values in the scf file (in kGauss) as: $\sigma_{c}[\text{ppm}] = -HFF * 1000$ (for a 100 T field, mind the minus sign).

You can also get the spin-susceptibility from the total induced magnetic spin moment (:MMTOT) using:

$$\chi_s[10^6 \text{cm}^3 \text{mol}^{-1} \text{cell}^{-1}] = \frac{M[\mu_B]}{\text{Volume}[\text{bohr}^3] * B[T]} * 6.258116 \quad (5.1)$$

which could be added to the orbital susceptibility to get the total molar susceptibility.

Another contribution, which is usually small, but can sometimes be large (in very anisotropic cases) is the spin-dipolar contribution [Laskowski et al., 2017]. After a converged scf calculation including the magnetic field (as above), copy case.indm from $\$WIENROOT/\$SRC/templates, select the proper atom and l=0-3 and change the last line (r-index, (l,s)index) to "3 5". Then run x lapwdm -up/dn . . . . From the difference of the total :XOP0xx values in the case.scfdmup/dn files (in T) you get the spin dipolar contribution $\sigma_{sd}[\text{ppm}]$ by multiplication with -10000 (for a field of 100 T, mind the minus sign).

Of course, also the orbital contribution is non-negligible in metallic compounds and one should also calculate the chemical shift (see above). Run x_nmr -metal [-p] [-noxi] [-quota yyy]. Please note: you will usually need an ENOURMOUS k-mesh (more than 50000 k-points), and also check convergence with respect to the fermi smearing 0.00x parameter. In case of diskspace problems use -quota yyy, so that the partial vector files contain only yyy k-points. The -noxi parameter excludes the contributions from both, the spin and orbital part of the macroscopic susceptibility, because we have found that this quantity is in most cases still an order of magnitude more difficult to converge, but gives only a small contribution to the shielding (a few ppm). In principle you could run x_nmr -noinit -metal -kbT 0.00x -xionly and check the corresponding susceptibility in case.xim. If you can reach convergence, you could add its contribution in the final integration using x_nmr -mode integ.

### 5.7 Wannier functions (wien2wannier)

This program was contributed by:

Wien2Wannier by J.Kunes. P.Wissgott and E.Assmann. Please cite the following paper when using it:


http://wien2wannier.github.io/

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

wien2wannier is an interface program between WIEN2k and WANNIER90 (http://www.wannier.org/) to obtain maximally localized Wannier functions from WIEN2k calculations. It provides the necessary overlap matrices for the construction of Wannier functions and besides some auxilliary programs it also contains a package for plotting the resulting Wannier functions.
5.7. WANNIER FUNCTIONS (WIEN2WANNIER) in real space. With this interface, the “whole world of WANNIER90”, i.e. applications which rely on maximally localized Wannier functions and the resulting hopping parameters (Transport, Berry phases (see Chapter 5.8), DMFT) can be combined with WIEN2k. WANNIER90 must be installed separately from http://www.wannier.org/ and should be cited when using it [Mostofi et al., 2008].

5.7.1 Usage

This section contains only a very brief summary of wien2wannier. Please consult the detailed wien2wannier_usersguide.pdf for more details, which is available from $WIENROOT/SRC/w2w or the “textbooks” site at http://www.wien2k.at. For a quick reference, see also the plain-text file CHEATSHEET in $WIENROOT/SRC/w2w.

Preparatory steps

Before running wien2wannier, one needs a converged WIEN2k calculation. Additionally, during the setup for wien2wannier, the bands which are to be taken into account will have to be specified, and the main character (e.g., d bands on atom 2) of these bands should be known. To obtain this information, a combination of partial DOS and bandstructure, or a band character plot is often necessary (e.g. spaghetti fat bands option, or SpaghettiPrimavera and prima.py, available in the unsupported software section of the WIEN2k website).

- Converge a Wien2k calculation: run[sp|sp_c] OPTIONS
- obtain band structure and partial DOS
- identify target bands and band characters

Then create a subdirectory with the necessary files using:

- prepare_w2wdir TARGET

which also gets the Fermi energy from case.scf (or case.scf2, if case.scf is not present (take care after x lapw2 -qtl -band!)) and change into this new directory TARGET.

Interface and Wannierization

- init_w2w [-up|-dn] generates various input files and performs the following steps:
  - x kgen -fbz Prepares an unshifted k-mesh in the full BZ. Of course, the mesh-density influences the quality of localization of the Wannier functions.
  - x findbands: looks in case.output1 for bands in a given energy range $[E_{\text{min}}; E_{\text{max}}]$ (in eV with EF=0), and outputs the corresponding band indices $b_{\text{min}}; b_{\text{max}}$. To choose the energy window of interest, consult the (partial) DOS and/or a band structure plot.
  - write_inwf: prepares the main input file case.inwf for the interface. The band indices $b_{\text{min}}; b_{\text{max}}$ have to be specified, and initial projections $A_{\text{mn}}$ may be given in terms of atomic sites and appropriate spherical harmonics.
  - write_win writes the input file case.win for wannier90.x on the basis of case.inwf and other files.
  - x wannier90 -pp reads the k-mesh in case.win and writes a list of nearest-neighbor k-points to case.nnkp.
- x lapw1 OPTIONS: computes the eigenvectors on the full-BZ k-mesh
  - you may use .machines and -p, -up/-dn, -orb,
– you may also consider spin-orbit: x lapwso OPTIONS

▶ x w2w [-up|-dn] [-p] [-so]: computes the overlaps $M_{mn}(k, b)$, initial projections $A_{mn}(k)$ and eigenvalues $E_n$, and writes them to case.mmn, case.amn, and case.eig.

▶ x wannier90 [-up|-dn] [-so]: computes the $U_{mn}(k)$ by maximum localization. Output is stored in case.wout.

Verification and Postprocessing

After a successful WANNIER90 run, one should check if the centers and spreads of the Wannier functions (printed in case.wout) are sensible. Another important consistency check is to compare the Wannier-interpolated bandstructure to the one computed by WIEN2k. wien2wannier also provides programs to create a real-space plot of the Wannier functions.

▶ compare band structures:
  With the option “hr.plot=T” in case.win, WANNIER90 writes a bandstructure derived from the Wannier-interpolated Hamiltonian $H(k)$ to case.band.dat. To compare this to the bandstructure computed by spaghetti, you can use gnuplot, using the command (including a conversion from Bohr to Å)
  
  gnuplot case.band.dat \n  p ‘case.spaghetti_ene’ u ($4+1.89):5, ’case_band.dat’ w l

The steps for plotting of Wannier functions are:

▶ write_inwplot: asks for a real-space grid on which the Wannier functions should be plotted, and writes case.inwplot.

▶ x wplot -wf m [-up|-dn][-p] [-so] evaluates Wannier function number m on the real-space grid, and writes the density $|w_m(r)|^2$ to case.m.psink and the phase arg $w_m(r)$ to case.m.psiarg.

▶ use positions from case.wout

▶ wplot2xsf converts all case*.psink and case*.psiarg files in the directory to the corresponding xsf files which can be opened by XCrySDen. It can also shift the origin according to case.centres.xyz.

▶ xcrysden --xsf case.m.xsf (or VESTA) visualizes the Wannier functions. Pick Tools > Data Grid from the menu and press OK. In the isosurface controls window choose an appropriate isovalue, e.g. 0.1, and check the Render +/- isovalue box.

5.7.2 Help and FAQ

Additional information about all programs can be accessed via the help flag, program -h.

And of course, read the detailed wien2wannier.userguide.pdf in $WIENROOT/SRC.w2w. In particular there is a FAQ section, which may answer your question.

5.8 Spontaneous Polarization, Piezoelectricity and Born Charges (BerryPI)

This program was contributed by:
5.8. SPONTANEOUS POLARIZATION, PIEZOELECTRICITY AND BORN CHARGES (BERRYPI)

These calculations are based on the “Modern Theory of Polarization” (Berry Phase) pioneered by [King-Smith and Vanderbilt, 1993, Resta et al., 1993], which noticed that (in a solid) we can only see a change of polarization \( \Delta P \) in response to an external perturbation, but not the polarization itself. BerryPI computes both, the ionic and electronic contributions to \( P \) using \texttt{wien2wannier} to obtain the overlap integral between two cell periodic parts of the Bloch functions. Of course, this theory applies only to insulators (semiconductors), but not for metals. For more details study the relevant literature (see the CPC paper mentioned above, which should be cited when this module is used in a publication) or the detailed tutorials at $WIENROOT/SRC/BerryPI/BerryPI$.

A current limitation of this implementation is that the structures must have orthogonal lattice vectors. This also means that cubic F and B centered lattices must be converted into a P-type conventional supercell with 4 (2) times as many atoms as the primitive cell. As this should be applied to insulators only, you have to use the TETRA method for the BZ integration. In addition, parallel execution is not yet supported directly (see below).

5.8.1 Options

The program is called using

\begin{verbatim}
berrypi -kNX:NY:NZ [-s -o -j -l] [-h]
\end{verbatim}

An online help of all options can be obtained with the \texttt{-h} switch.

The parameter \texttt{-kNX:NY:NZ} is mandatory and determines the k-mesh for the BZ sampling.

The additional switches \texttt{-s} allows spin-polarized calculations; \texttt{-o} supports additional orbital potentials (LDA+U or EECE); \texttt{-j} includes spin-orbit coupling (\texttt{x lapwso}); and \texttt{-l} skips the \texttt{lapw1} run.

The option \texttt{-l} can be used to make the most time consuming \texttt{lapw1} step running in parallel. Execute ‘’x kgen -fbz’’; ‘’x lapw1 -p’’ and ‘’join_vectorfiles case NUM’’ before the call fo \texttt{berrypi -l}.

5.8.2 Spontaneous Polarization

To obtain \( \Delta P \) one has to do two calculations, one for the “unperturbed structure (\( \lambda_0 \))” and one for the “perturbed one (\( \lambda_1 \))” and obtain \( \Delta P = P_1 - P_0 \).

Start out with the distorted structure (eg. the ferroelectric phase of BaTiO\(_3\)) and perform a standard WIEN2k calculation. Then run the berrypi program:

\begin{verbatim}
mkdir case;cd case;mkdir case0;mkdir casel
   # create suitable directories
   cd casel
   makestruct          # create your structure
\end{verbatim}
init_lapw -b ... # initialize wien2k
run_lapw ... # run scf cycle
berrypi -k6:6:6 # run berrypi

where -k defines a suitable k-mesh. This will give you the corresponding x,y,z components of the polarization as:

<table>
<thead>
<tr>
<th>Value</th>
<th>spin</th>
<th>dir(1)</th>
<th>dir(2)</th>
<th>dir(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic polarization (C/m²) sp(1)</td>
<td>[ -9.684673e-12, -2.406503e-13, 4.879618e-01]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ionic polarization (C/m²) sp(1)</td>
<td>[ 1.365657e-11, 1.365657e-11, -1.760570e-01]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tot.spin polariz.=Pion+Pel (C/m²) sp(1)</td>
<td>[ 3.971897e-12, 1.341592e-11, 3.119048e-01]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOTAL POLARIZATION (C/m²) both | [ 3.971897e-12, 1.341592e-11, 3.119048e-01] |

Now copy all files to the case0 directory, rename the files and change the struct file such that it corresponds to the undistorted (cubic) structure (keeping all other inputs identical):

cd ..;cp -r case1 case0;cd case0;rename_files case1 case0
edit case0.struct # create undistorted structure
x dstart # new starting density
run_lapw ... # run scf cycle
berrypi -k6:6:6 # run berrypi

The spontaneous polarization in z-direction is defined as the difference in z component of polarization between the non-centrosymmetric $P_z(\lambda_1)$ and centrosymmetric structure $P_z(\lambda_0)$. In this case $P_z(\lambda_0) = 0$ (output not shown) and the resultant spontaneous polarization is $P_s = 0.31$ C/m². Please consider the effects of possible π wrapping, so in general the smallest possible value should be considered. If there is a suspect of π-wrapping artifacts, it is useful to study intermediate structures (between $\lambda_1$ and $\lambda_0$) and ensure continuity in the evolution of $P_z$.

5.8.3 Born effective charges

The Born effective charge $Z^*_{s,\alpha\beta}$ of an atom $s$ is defined as the change in polarization due to a displacement of its position. These charges are also used to estimate the LO/TO splitting of the optical vibrational modes at $\Gamma$.

For the calculation of the Born effective charge of As in GaAs one has first to create a “P”-type supercell with 4 formula units/cell (see limitations above). One of the 4 As atoms has to be displaced along the z-axis from it’s equilibrium position by +0.01($\lambda_1$) and -0.01($\lambda_2$) in fractional coordinates. Then perform (identical) WIEN2k calculations for the two structures and run berrypi -k6:6:6. The two calculations yield lines like:

"lambda1"
ELECTRONIC POLARIZATION

<table>
<thead>
<tr>
<th>Value</th>
<th>spin</th>
<th>dir(1)</th>
<th>dir(2)</th>
<th>dir(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berry phase (rad) [-pi ... +pi] up+dn</td>
<td>[ 3.151667e-10, -2.544453e-09, -1.081339e+00]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electronic polarization (C/m²) sp(1)</td>
<td>[ 2.441627e-11, -1.971212e-10, -8.377237e-02]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IONIC POLARIZATION

<table>
<thead>
<tr>
<th>Elem.</th>
<th>Fractional coord.</th>
<th>spin</th>
<th>val</th>
<th>dir(1)</th>
<th>dir(2)</th>
<th>dir(3)</th>
</tr>
</thead>
</table>
...
5.9. GETTING ON-LINE HELP

"lambda2"

ELECTRONIC POLARIZATION

<table>
<thead>
<tr>
<th>Value</th>
<th>spin</th>
<th>dir(1)</th>
<th>dir(2)</th>
<th>dir(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Berry phase (rad) [-pi ... +pi] up+dn</td>
<td>7.675118e-10, -2.577606e-09, 1.081339e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electronic polarization (C/m²) sp(1)</td>
<td>5.945987e-11, -1.996896e-10, 8.377237e-02</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IONIC POLARIZATION

<table>
<thead>
<tr>
<th>Elem.</th>
<th>Fractional coord.</th>
<th>spin</th>
<th>val</th>
<th>dir(1)</th>
<th>dir(2)</th>
<th>dir(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total ionic phase wrap. (rad) sp(1)</td>
<td>3.686359e-09, 3.686359e-09, -9.424778e-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ionic polarization (C/m²) sp(1)</td>
<td>2.855857e-10, 2.855857e-10, -7.301465e-02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The total (ionic + electronic) phase along z-axis in the case of "\( \lambda_1 \)" and "\( \lambda_2 \)" is -0.13886 and 0.13886 rad, respectively. The Born charge can be obtained from these phases \( \phi \) as

\[
Z_{zz}^* = \frac{1}{2\pi} \frac{\delta \phi_z}{\delta \rho_z}
\]  

(5.2)

where \( \delta \rho \) is the relative displacement (0.02) in fractional coordinates. The calculation yields \( Z_{zz}^* = -2.18 \). The negative sign is indicative of a higher electronegativity of As as compared to that for Ga. Please consider the effects of possible \( \pi \) wrapping, so in general the smallest possible value should be considered.

5.8.4 Piezoelectric constants

For such calculations you need to calculate the Berry phases for the reference (equilibrium) structure (e.g. the tetragonal ferroelectric PbTiO\(_3\) structure) and a perturbed structure, where a compressive strain \( \epsilon_z \) of 0.1 % has been applied in the z-direction (for the latter structure one should also perform a new optimization of the internal coordinates).

The piezoelectric coefficient \( \epsilon_{zz} \) is defined as change in polarization with respect to the applied strain:

\[
\epsilon_{zz} = \frac{dP_z}{d\epsilon_z}
\]  

(5.3)

5.9 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 user 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 whether 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.

## 5.10 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.

### 5.10.1 eplot

The script eplot_lapw plots total energy vs. volume or total energy vs. c/a-ratio or b/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”. Alternatively you can use eplot_lapw -a search-string-in-scf-files, which generates case.outputeos.meshp, eos.meshp1 and deos1 files. These files allow for a comparison of the Gibbs energy as function of pressure for the two different phases.

A typical sequence to determine the transition pressure of this phase transition (assuming that the struct files and initializations have been done before) would look like:

```bash
eplot_lapw
```

### 5.10.2 gibbs

The script gibbs_lapw (provided by M. Jamal) is an extension of eplot_lapw and can also plot Volume vs. Pressure curves as well as the Gibbs energy difference (stored in case.outputDeltaG) of two different phases as function of Pressure.

When interested in pressure driven phase transitions, one can do calculations for the two phases of interest in two different subdirectories and perform “Volume optimization” (using x optimize; optimize.job, see sec. 5.3). Once this has been finished, one can use gibbs_lapw (instead of eplot_lapw), which will also create case.outputeos.meshp, eos.meshp1 and deos1 files. These files allow for a comparison of the Gibbs energy as function of pressure for the two different phases.

A typical sequence to determine the transition pressure of this phase transition (assuming that the struct files and initializations have been done before) would look like:
5.10. INTERFACE SCRIPTS

```bash
cd dir1
x optimize # select Volume optimization and a suitable volume range
optimize.job # eventually change some run or save-options before
gibbs.lapw -v vol
cd ../dir2
x optimize # select Volume optimization and a suitable volume range
optimize.job # eventually change some run or save-options before
cp ../dir1/eos.meshp1 eos.meshp2
cp ../dir1/deos1 deos2
gibbs.lapw -v vol
```

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

```
gibbs.lapw -h
```

which will yield:

```
gibbs.lapw [-v vol] [-a string_in_scf-files] [-plt/-gbs/-ene] [-2D]
```

For instance, `gibbs.lapw -a pbe` will analyse all scf files ‘*pbe.scf’.

5.10.3 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

```
```

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

5.10.4 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. It can be used to generate all partial DOS plots in a simple way to get an overview. 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
```

5.10.5 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 supports also the SUM-DOS option (see description of TETRA).

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

For a description of how to use the script for batch like execution call the script using
You can also use the script `dosplot_all.lapw [-up]` to generate default-plots (4 lines per plot) of all partial DOS cases as defined in case.int.

### 5.10.6 Cgrace_lapw, Cgrace_conf_lapw and Cgrace_dos_lapw

These three scripts can be used to plot x,y-data using `xmgrace`. Specifically

- **Cgrace_lapw** generates a “grace-file” `IRgrace.agr` which can be plotted using `xmgrace IRgrace.agr` using an intermediate `IRgrace.inf`, produced by either `Cgrace_dos_lapw` or `Cgrace_conf_lapw` or for DOS-plotting `Cgrace_lapw -dos [-up/-dn -eV/-Ryd]`.
- **Cgrace_dos_lapw** lets you select which partial-DOS curves you want to plot and generates a “grace-file” `IRgrace.agr` which can be plotted using `xmgrace IRgrace.agr`. It supports also the SUM-DOS option (see description of TETRA). It produces first `IRgrace.inf` and with this configuration file `Cgrace_lapw` makes the actual grace-file.
- **Cgrace_conf_lapw** can produce (or modify) `IRgrace.inf`. You can plot any x,y-data file (with multiple columns and headers) like the files produced by tetra, optics, xspec, ...

For DOS-plotting call it as `Cgrace_conf_lapw -dos [-up/-dn]` You can configure multiple frames, and in each frame define several curves for plotting.

For spinpolarized cases you must call the scripts with `-up/-dn` options.

For a description of how to use the scripts call the scripts using `-h` switch.

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

### 5.10.7 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).

### 5.10.8 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
```

### 5.10.9 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`.**
5.10. INTERFACE SCRIPTS

5.10.10 prepare.xsf.lapw

This program was contributed by:

David Koller
Institute for Materials Chemistry
TU Vienna
koller@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.

The script `prepare.xsf.lapw` produces 3D data of the electron density (or the potential) in XCrystDen-format (`case.xsf`) for plotting with XCrystDen (Menu → Tools → Data Grid). It is written in Python and also uses the programs `lapw5` and `str2xcr.exe` (included in the WIEN2k distribution).

It requires an input file `case.inxsf`:

```
# This is an inxsf-file
>D9  clmval  # unit # 9 in def-file
>D1  clmvaldn # unit # 11 in def File

>IS # Start of end part of in5-file
RHO
ATU VAL NODEBUG # careful VAL/TOT!!!
NONORTHO
>IE # closes what was started with >IS

>C0 0 0 0  # Start-Corner of part of unit cell (compared to lattice vectors of conventional cell)
>CX 0.1 0.1 0 # x-end
>CY 0.2 0.2 1 # y-end
>CZ 0.5 0.5 0  # z-end
# use for fcc:
#>C0 0 0 0
#>CX 0.5 0.5 0
#>CY 0.5 0.5 0
#>CZ 0.5 0.5 0
# entire cell:
#>C0 0 0 0
#>CX 1 0 0
#>CY 1 0 0
#>CZ 1 0 0

>NX 30 # number of data points in x-direction
>NY 30
>NZ 30
>IZ 3 2 3 # additional cells in in5-file

>PS # parallel start
machine1
machine1
machine2
>PE # parallel end
# >PM

# End of inxsf-file
```

In this file comments are designated by ‘#’. Markers at the beginning of a line consisting of ‘>’ followed by two characters determine the content of this line or of the following lines, depending on the marker.

Explanation of the markers:

>`D9`: The suffix of the main data file. It corresponds to unit 9 in the file `lapw5.def`
>D1: The suffix of a second data file which can be optionally added to or subtracted from the main data file. It corresponds to unit 11 in the file lapw5.def.

>IS: This starts a section which needs to be closed with ‘>IE’. The lines between these two markers will be used as lines 6-8 in the in5-file.

>IZ: This will be used as line 4 in the in5-file.

>C0: Coordinates of a corner of a three-dimensional box, delimited by parallel planes, in which the data should be plotted. The units of these numbers are the unit vectors of the conventional cell (e.g. 0.5 0.5 0 is the centre of the xy-plane which would be the 1d-position in space group 111).

>CX: Coordinates of the x-end corner of the box

>CY: Coordinates of the y-end corner of the box

>CZ: Coordinates of the z-end corner of the box

>NX: Number of data points in x-direction

>NY: Number of data points in y-direction

>NZ: Number of data points in z-direction

>PS / >PE / >PM: determine the parallelization

This script also contains support for parallel execution. One possibility is to include ‘>PM’. In this case the file .machines is used to determine which hosts are used. More details can be found in the section about parallel WIEN2k. If ‘>PM’ is not present (or commented) it is possible to specify the desired hosts between ‘>PS’ and ‘>PE’. If neither ‘>PM’ nor ‘>PS’ are present, the script will be executed in non-parallel way which should work well enough in most cases.

5.10.11 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

5.10.12 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 (ε1) is not a simple additive quantity concerning the spin (see [Ambrosch-Draxl and Sofo, 2006]). The KK transformation should then be done non-spinpolarized (x kram) resulting in files: case.epsilon, case.eloss, case.sumrules or case.sigmak.

This script can also be “missused” to add or subtract (add the keyword “sub”) the content of case.jointup and case.jointdn, when they come from calculations of different band-ranges, ...
6 Programs for the initialization

Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 NN</td>
<td>nearest neighbor distances</td>
<td>103</td>
</tr>
<tr>
<td>6.2 SGROUP</td>
<td></td>
<td>104</td>
</tr>
<tr>
<td>6.3 SYMMETRY</td>
<td></td>
<td>104</td>
</tr>
<tr>
<td>6.4 LSTART</td>
<td></td>
<td>105</td>
</tr>
<tr>
<td>6.5 KGEN</td>
<td></td>
<td>107</td>
</tr>
<tr>
<td>6.6 DSTART</td>
<td></td>
<td>108</td>
</tr>
</tbody>
</table>

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 “inaccurate” 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 useful 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 [Desclaux, 1969, Desclaux, 1975] 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 $3d^94s^2$).

The program produces “WARNINGS” if $R_0$ 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 specified 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-functional switch. Currently XC_LDA (LDA, [Perdew and Wang, 1992]) as well as three GGAs, XC_PBE [Perdew et al., 1996], XC_WC [Wu and Cohen, 2006] and XC_PBESOL [Perdew et al., 2008] are supported by lstart.

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,GOUP; = 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 useful 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 \( \kappa = -s(j + 1/2) \)

<table>
<thead>
<tr>
<th>( j = l + s/2 )</th>
<th>( s = +1 )</th>
<th>( s = -1 )</th>
<th>( s = +1 )</th>
<th>( s = -1 )</th>
<th>( s = +1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s = 0 )</td>
<td>1/2</td>
<td>1/2</td>
<td>-1</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>( p )</td>
<td>1/2</td>
<td>3/2</td>
<td>1</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td>( d )</td>
<td>3/2</td>
<td>5/2</td>
<td>2 -3</td>
<td>-5</td>
<td>4</td>
</tr>
<tr>
<td>( f )</td>
<td>5/2</td>
<td>7/2</td>
<td>3</td>
<td>-4</td>
<td>6</td>
</tr>
</tbody>
</table>

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 length 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 128.

6.5.1 Execution

The program kgen is executed by invoking the command:

    kgen kgen.def or x kgen [-so -fbz -hf]

With the switch -so it uses a file case.ksym (usually generated by symmetso) instead of case.struct and does not add inversion symmetry. The -fbz switch generates a k-mesh in the full Brillouin zone (no symmetry).

6.5.2 Dimensioning parameters

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDKP</td>
<td>number of inequivalent k-points (like NKPT in other programs)</td>
</tr>
<tr>
<td>NWX</td>
<td>internal parameter, must be increased for very large k-meshes</td>
</tr>
<tr>
<td>INDEXM</td>
<td>internal parameter, must be increased for very large k-meshes</td>
</tr>
</tbody>
</table>

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. You may also specify a larger LUSE value in case.in0 (default LUSE=7) for the angular integration. 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 -p]

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.rspicore (this is activated during scf when a .lcore file is present. It can run in mpi-parallel mode (-p) for big cases (typically more than 20 atoms) and core-superposition.
6.6.2 Dimensioning parameters

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPT</td>
<td>number of ( r )-mesh points in atomic density (should be the same as in LSTART)</td>
</tr>
<tr>
<td>LMAX2</td>
<td>max ( l ) in LM expansion</td>
</tr>
<tr>
<td>NCOM</td>
<td>number of LM terms in density</td>
</tr>
</tbody>
</table>
7 Programs for running an SCF cycle

Contents

7.1 LAPW0 .................................................. 111
7.2 DFTD3 .................................................. 115
7.3 NLVDW .................................................. 119
7.4 ORB ...................................................... 120
7.5 LAPW1 .................................................... 124
7.6 HF ........................................................ 128
7.7 LAPWSO .................................................. 130
7.8 LAPW2 .................................................... 133
7.9 SUMPARA ................................................ 137
7.10 LAPWDM .................................................. 138
7.11 LCORE .................................................... 139
7.12 MIXER .................................................... 141

In sections 7.1-7.12 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 \( V_{\text{tot}} \) 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, 1981]. 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 Gauss-Legendre integration is used to reproduce the potential using a lattice harmonics representation. 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 \( \rho \ast V \) is calculated according to the formalism of [Weinert et al., 1982].

The Hellmann-Feynman force contribution to the total force is also calculated [Yu et al., 1991].

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:

\[
\text{lapw0 lapw0.def or } x \text{ lapw0 [ } -p -eece -grr -nlvdw]}
\]

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 \cdot \text{gt.} \{\text{number of } l,m \text{ with } m = 0 \} + [2 \ast \text{number of } l,m \text{ 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:

\[
\text{------------------ top of file: case.in0 --------------------}
\text{TOT XC_PBE ! MULT/COUL/EXCH/POT /TOT ; VXC-SWITCH}
\text{NR2V IFFT 13 ! R2V EECE/HYBR IFFT LUSE}
\]

\text{------------------ top of file: case.in0 --------------------}
\text{TOT XC_PBE ! MULT/COUL/EXCH/POT /TOT ; VXC-SWITCH}
\text{NR2V IFFT 13 ! R2V EECE/HYBR IFFT LUSE}
Interpretive comments follow:

**line 1:** free format

```plaintext
switch, indxc, xc1, xc2, xc3
```

- **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 (Ekin and EXC in `case.scf`).
- **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

**indxc**

One keyword (XC_NAME), four keywords (EX_NAME EC_NAME VX_NAME VC_NAME) or `LIBXC`-keyword(s) to specify the XC-energy/potential. The description for the onsite and full hybrid functionals are in sections 4.5.7 and 4.5.8, respectively.

**XC_NAME**

Global keyword to specify the X and C energy and potential. Some of the most popular options are (for all options see Table 7.3 or in `SRC/lapw0/vxclm2.f`):

- **XC_LDA** : parametrization [Perdew and Wang, 1992] of accurate Monte-Carlo data of the homogeneous electron gas
- **XC_PBE** : GGA PBE [Perdew et al., 1996]
- **XC_WC** : GGA WC [Wu and Cohen, 2006, Tran et al., 2007]
- **XC_PBESOL** : GGA PBEsol [Perdew et al., 2008]
- **XC_SCAN** : probably the best meta-GGA (energy functional only, uses PBE for the potential) up to now [Sun et al., 2015b]. In order to generate the required `case.vresp*` files, you need `case.inm.vresp` (cp `WIENROOT/SRC/templates/template.inm.vresp` case.inm.vresp and run one scf cycle with XC_PBE after creation of `case.inm.vresp`. Only afterwards change indxc to XC_SCAN. Meta-GGA functionals may eventually require a larger FFT factor (`case.in0`) or GMAX (`case.in2`) than for GGA functionals.
- **XC_MBJ** : modified Becke-Johnson (mBJLDA) potential $V_{XC}$ [Tran and Blaha, 2009]. Uses the mBJ-exchange + LDA-correlation potential and yields band gaps in very good agreement with experiment. By default, the xc-energy $E_{XC}$ is from LDA. For detailed usage see chapter about mBJ calculations (4.5.10).

**EX_NAME EC_NAME VX_NAME VC_NAME**

Keywords for $E_X$ (Table 7.4), $E_C$ (Table 7.5), $V_X$ (Table 7.6) and $V_C$ (Table 7.7)
Keywords to use functionals from the library of exchange and correlation functionals \texttt{LIBXC} (\url{http://octopus-code.org/wiki/Libxc}). A few points about the use of \texttt{LIBXC}:

- \texttt{LIBXC \cite{Marques2012}} is a separate program that has to be downloaded and compiled with the same compiler as the one used for \texttt{WIEN2k} (see 11.1.2).
- One (\texttt{X}, \texttt{C}, or \texttt{XC}) or two (\texttt{X} and \texttt{C}) \texttt{LIBXC} keywords can be specified. The full list of possible keywords is in \texttt{$\text{WIENROOT/SRC\_lapw0/xc\_funcs.h}$} or on the \texttt{LIBXC} website.
- It is possible to combine a functional/potential from \texttt{LIBXC} with a functional/potential from \texttt{WIEN2k} by also specifying one or several of the keywords \texttt{EX\_NAME}, \texttt{EC\_NAME}, \texttt{VX\_NAME}, \texttt{VC\_NAME} (but not \texttt{XC\_NAME}) which will overrule the corresponding \texttt{LIBXC} choice (except if \texttt{NAME}=\texttt{NONE}).
- A few \texttt{LIBXC} keywords (\texttt{XC\_GGA\_X\_LB}, \texttt{XC\_GGA\_X\_LBM}, \texttt{XC\_MGGA\_X\_BJ06}, \texttt{XC\_MGGA\_X\_TB09}, \texttt{XC\_MGGA\_X\_RPP09}) correspond to only a potential (the corresponding energy is zero). However (see the point just above), it is possible to specify an energy functional with \texttt{EX\_NAME} and \texttt{EC\_NAME}.
- As for the meta-GGA energy functionals called with the \texttt{WIEN2k} keywords (see table 7.3), the default associated potential is PBE.
- Note that the possibility offered by \texttt{LIBXC} to specify the value of parameters in several functionals has not been implemented in \texttt{WIEN2k}.

Example for PBE with the

- global keyword : \texttt{XC\_PBE}
- individual keywords : \texttt{EX\_PBE EC\_PBE VX\_PBE VC\_PBE}
- \texttt{LIBXC}-keywords : \texttt{XC\_GGA\_X\_PBE XC\_GGA\_C\_PBE}

\texttt{xc1,xc2,xc3} optional input(s) for certain XC options:

Example for MGGA\_MS2: \texttt{XC\_MGGA\_MS 0.504 0.14601 4.0}
\texttt{XC\_LDA} or \texttt{XC\_PBE}: to modify the spin scaling (reduction of spin-polarization) according to \cite{Ortenzi2012}. \texttt{xc1} must be between 0 and 2.

\textbf{line 2:} free format (only blanks are allowed as separator)

\texttt{RPRINT, H-mod, FFTopt, LUSE}

\texttt{RPRINT NR2V} no additional output
7.2. **DFTD3**

**R2V**  Exchange-correlation (\texttt{case.r2v}), Coulomb (\texttt{case.vcoul}) and total potentials (\texttt{case.vtotal}) are written as \((r^2V)\) to a file for plotting with \texttt{lapw5} (cp \texttt{case.vtotal case.clmval}; use “VAL” for normalization in \texttt{case.in5})

**H-mod**  

**EECE**  Onsite Hartree-Fock (inside spheres) for selected electrons (see 4.5.7)

**HYBR**  Onsite 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 \texttt{runeece_lapw})

**LUSE**  optional l-max value for the angular grid used in \texttt{xcpot1}. For standard LDA/GGA the minimal value is max L value of LM-list in case.in2 + 2; the default is 13 and for EECE one should use a better, antialiased grid, thus a large negative LUSE-value is recommended (and set automatically by \texttt{runeece_lapw})

**line 3:**  free format (must be omitted when IFFT is not specified above)

\texttt{IFFTx, IFFTy, IFFTz, IFFTfactor, iprint}

**IFFTx,\texttt{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 \texttt{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. \texttt{iprint=0} will greatly reduce case.output0 (in particular for \texttt{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 [Stahn et al., 2001]):

**line 4:**  free format

\texttt{IFIELD, EFIELD, WFIELD}

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

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

**WFIELD**  optional value for lambda (see output of \texttt{IFIELD=}\texttt{-999}).

### 7.2  DFT-D3 (Calculate the dispersion energy with DFT-D3)

\texttt{dftd3} calculates the dispersion energy and forces using the semi-empirical DFT-D3 method of [Grimme et al., 2010]. Since this method depends only on the positions of atoms (no dependence on the electron density) it is very fast and adds very little computer time. The \texttt{dftd3} package is not included by default in \texttt{WIEN2k}, but can be downloaded from the website of the group of S. Grimme \url{http://www.thch.uni-bonn.de/tc/index.php}. When compilation is done, the executable \texttt{dftd3} has to be copied in the \texttt{$WIENROOT} directory.
### Table 7.3: global XC-switches

<table>
<thead>
<tr>
<th>Keyword for ( E_X )</th>
<th>Type</th>
<th>Reference</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX_NONE</td>
<td>LDA</td>
<td>[Kohn and Sham, 1965]</td>
<td>no exchange energy</td>
</tr>
<tr>
<td>EX_LDA</td>
<td>LDA</td>
<td>[Kohn and Sham, 1965]</td>
<td>screened LDA for hybrid (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_SDLA</td>
<td>LDA</td>
<td>[Tran and Blaha, 2011]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_B88</td>
<td>GGA</td>
<td>[Becke, 1988]</td>
<td>semilocal part of hybrid B3LYP</td>
</tr>
<tr>
<td>EX_PW91</td>
<td>GGA</td>
<td>[Perdew et al., 1992]</td>
<td>xc1 (( \alpha )) is required</td>
</tr>
<tr>
<td>EX_P86x</td>
<td>GGA</td>
<td>[Engel and Vosko, 1991]</td>
<td>x2 (( \omega )) and xc3 (( \beta )) are required</td>
</tr>
<tr>
<td>EX_BP86</td>
<td>GGA</td>
<td>[Perdew et al., 1996]</td>
<td>screened PBE for hybrid (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_REVPB86</td>
<td>GGA</td>
<td>[Zhang and Yang, 1998]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_HCTH93</td>
<td>GGA</td>
<td>[Hammer et al., 1999]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_HCTH120</td>
<td>GGA</td>
<td>[Boese et al., 2000]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_HCTH147</td>
<td>GGA</td>
<td>[Boese et al., 2000]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_HCTH407</td>
<td>GGA</td>
<td>[Boese and Handy, 2001]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_AM05</td>
<td>GGA</td>
<td>[Armiento and Mattsson, 2005]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_WC</td>
<td>GGA</td>
<td>[Wu and Cohen, 2006]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_PBE-ALPHA</td>
<td>GGA</td>
<td>[Madsen, 2007]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_PBE-SOL</td>
<td>GGA</td>
<td>[Perdew et al., 2008]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_SOGGA</td>
<td>GGA</td>
<td>[Zhao and Truhlar, 2008]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_BG02</td>
<td>GGA</td>
<td>[Ruzsinszky et al., 2009]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_PBEINT</td>
<td>GGA</td>
<td>[Fabiano et al., 2010]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_OPTPB86</td>
<td>GGA</td>
<td>[Klimel et al., 2010]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_OPTTB08</td>
<td>GGA</td>
<td>[Klimel et al., 2010]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_HTB8</td>
<td>GGA</td>
<td>[Klimel et al., 2011]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_AK13</td>
<td>GGA</td>
<td>[Armiento and Kummel, 2013]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_CAP</td>
<td>GGA</td>
<td>[Carmona-Espinola et al., 2015]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_SG4</td>
<td>GGA</td>
<td>[Constantin et al., 2016]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_MPBE</td>
<td>GGA</td>
<td>[Haas et al., 2010]</td>
<td>semilocal part of hybrid B3PW91</td>
</tr>
<tr>
<td>EX_RSPW91</td>
<td>GGA</td>
<td>[Becke, 1993]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_B3LYP</td>
<td>GGA</td>
<td>[Boese et al., 1999]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_TIPS</td>
<td>GGA</td>
<td>[Henderson et al., 2008]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_REVTIPS</td>
<td>GGA</td>
<td>[Weintraub et al., 2009]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_HSHB86</td>
<td>GGA</td>
<td>[Henderson et al., 2008]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_SHSB89X</td>
<td>GGA</td>
<td>[Henderson et al., 2008]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_QVXC</td>
<td>MGGA</td>
<td>[Van Voorhis and Scuseria, 1998]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_PKZB</td>
<td>MGGA</td>
<td>[Perdew et al., 1999]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_TPPS</td>
<td>MGGA</td>
<td>[Henderson et al., 2008]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_REVTTPS</td>
<td>MGGA</td>
<td>[Weintraub et al., 2009]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_HSHMPBE</td>
<td>MGGA</td>
<td>[Henderson et al., 2008]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_SHSB89X</td>
<td>MGGA</td>
<td>[Henderson et al., 2008]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_QVXC</td>
<td>MGGA</td>
<td>[Van Voorhis and Scuseria, 1998]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_PKZB</td>
<td>MGGA</td>
<td>[Perdew et al., 1999]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
<tr>
<td>EX_TPPS</td>
<td>MGGA</td>
<td>[Henderson et al., 2008]</td>
<td>average of ( \langle \nabla \rho \rangle / \rho ) for \texttt{mB8} (\texttt{case.in0.grp})</td>
</tr>
</tbody>
</table>

Table 7.4: EX-switches
### 7.2.1 Execution

The program **dftd3** is executed by invoking the command:

```
  x dftd3
```

#### 7.2.2 Input

The options for the **dftd3** package have to be specified in the input file `case.indftd3`. If no input file is created by the user, then the script `run(sp)_lapw` will automatically copy the default one (which is the recommended one) from `$WIENROOT/SRC/templates/`:

```
---------------- top of file: case.indftd3 ----------------
method bj
func default
grad yes
pbc yes
abc yes
cutoff 55
cnthr 40
num no
---------------- bottom of file: ----------------------
```

A short summary of the options is given below and more details can be found in the file **man.pdf** included in the **dftd3** TAR file. Note that `case.indftd3` is read by the C-SHELL script and that all data should be written in small letters.

- **method**: choice of the DFT-D method: `bj` (the recommended one), `zero` or `old` (this leads to the older DFT-D2 method).
- **func <functional>**: three choices are possible: (a) `default`, which means the functional specified in `case.in0` (possible only if the parameters $s_6$, $s_8$, etc. for this functional are available), (b) one of the functionals listed in the FORTRAN file `dftd3.f` (e.g., `pbe` or `b-lyp`) or (c) `none`, which means that the parameters $s_6$, $s_8$, etc. are read from the file `.dftd3par.hostname` created by the user in his home (not working) directory. Note that for hybrid functionals, it is mandatory to specify the functional name (`default` will not work).
<table>
<thead>
<tr>
<th>Keyword for $V_X$</th>
<th>Type</th>
<th>Reference</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VX_LDA</td>
<td>LDA</td>
<td>[Kohn and Sham, 1965]</td>
<td>no exchange potential</td>
</tr>
<tr>
<td>VX_SLDA</td>
<td>LDA</td>
<td>[Tran and Blaha, 2011]</td>
<td></td>
</tr>
<tr>
<td>VX_B88</td>
<td>GGA</td>
<td>[Becke, 1988]</td>
<td></td>
</tr>
<tr>
<td>VX_PW91</td>
<td>GGA</td>
<td>[Perdew et al., 1992]</td>
<td></td>
</tr>
<tr>
<td>VX_EX93</td>
<td>GGA</td>
<td>[Engel and Vosko, 1993]</td>
<td></td>
</tr>
<tr>
<td>VX_LB94</td>
<td>GGA</td>
<td>[van Leeuwen and Barerends, 1994]</td>
<td></td>
</tr>
<tr>
<td>VX_PBE</td>
<td>GGA</td>
<td>[Perdew et al., 1996]</td>
<td></td>
</tr>
<tr>
<td>VX_REVPE</td>
<td>GGA</td>
<td>[Zhang and Yang, 1998]</td>
<td></td>
</tr>
<tr>
<td>VX_RPBE</td>
<td>GGA</td>
<td>[Hammer et al., 1999]</td>
<td></td>
</tr>
<tr>
<td>VX_HCTH93</td>
<td>GGA</td>
<td>[Hamprecht et al., 1998]</td>
<td></td>
</tr>
<tr>
<td>VX_HCTH120</td>
<td>GGA</td>
<td>[Boese et al., 2000]</td>
<td></td>
</tr>
<tr>
<td>VX_HCTH147</td>
<td>GGA</td>
<td>[Boese et al., 2000]</td>
<td></td>
</tr>
<tr>
<td>VX_HCTH407</td>
<td>GGA</td>
<td>[Boese and Handy, 2001]</td>
<td></td>
</tr>
<tr>
<td>VX_AM05</td>
<td>GGA</td>
<td>[Armiento and Mattsson, 2005]</td>
<td></td>
</tr>
<tr>
<td>VX_WC</td>
<td>GGA</td>
<td>[Wu and Cohen, 2006]</td>
<td></td>
</tr>
<tr>
<td>VX_PBE-ALPHA</td>
<td>GGA</td>
<td>[Madsen, 2007]</td>
<td></td>
</tr>
<tr>
<td>VX_PBEOSOL</td>
<td>GGA</td>
<td>[Perdew et al., 2008]</td>
<td></td>
</tr>
<tr>
<td>VX_SOOGA</td>
<td>GGA</td>
<td>[Zhuo and Truhlar, 2008]</td>
<td></td>
</tr>
<tr>
<td>VX_RGGE2</td>
<td>GGA</td>
<td>[Ruzsinszky et al., 2009]</td>
<td></td>
</tr>
<tr>
<td>VX_PBEEXT</td>
<td>GGA</td>
<td>[Fabiano et al., 2010]</td>
<td></td>
</tr>
<tr>
<td>VX_OPTPBE</td>
<td>GGA</td>
<td>[Klimes et al., 2010]</td>
<td></td>
</tr>
<tr>
<td>VX_OPTB88</td>
<td>GGA</td>
<td>[Klimes et al., 2010]</td>
<td></td>
</tr>
<tr>
<td>VX_OPTB88B</td>
<td>GGA</td>
<td>[Klimes et al., 2011]</td>
<td></td>
</tr>
<tr>
<td>VX_JHBS</td>
<td>GGA</td>
<td>[Haas et al., 2011]</td>
<td></td>
</tr>
<tr>
<td>VX_AK13</td>
<td>GGA</td>
<td>[Armiento and Kummel, 2013]</td>
<td></td>
</tr>
<tr>
<td>VX_CAP</td>
<td>GGA</td>
<td>[Carmona-Espinola et al., 2015]</td>
<td></td>
</tr>
<tr>
<td>VX_SG4</td>
<td>GGA</td>
<td>[Constantin et al., 2016]</td>
<td></td>
</tr>
<tr>
<td>VX_MPBE</td>
<td>GGA</td>
<td>[Haas et al., 2010]</td>
<td></td>
</tr>
<tr>
<td>VX_B3PW91</td>
<td>GGA</td>
<td>[Becke, 1993]</td>
<td></td>
</tr>
<tr>
<td>VX_BLYP</td>
<td>GGA</td>
<td>[Stephens et al., 1994]</td>
<td></td>
</tr>
<tr>
<td>VX_PBE</td>
<td>GGA</td>
<td>[Tran and Blaha, 2011]</td>
<td></td>
</tr>
<tr>
<td>VX_SWC</td>
<td>GGA</td>
<td>[Tran and Blaha, 2011]</td>
<td></td>
</tr>
<tr>
<td>VX_SPBESOL</td>
<td>GGA</td>
<td>[Tran and Blaha, 2011]</td>
<td></td>
</tr>
<tr>
<td>VX_SB88</td>
<td>GGA</td>
<td>[Tran and Blaha, 2011]</td>
<td></td>
</tr>
<tr>
<td>VX_SHSFBE</td>
<td>GGA</td>
<td>[Henderson et al., 2008]</td>
<td></td>
</tr>
<tr>
<td>VX_SHSFESOL</td>
<td>GGA</td>
<td>[Henderson et al., 2008]</td>
<td></td>
</tr>
<tr>
<td>VX_SHSFBB88</td>
<td>GGA</td>
<td>[Weintraub et al., 2009]</td>
<td></td>
</tr>
<tr>
<td>VX_SHSF897X</td>
<td>GGA</td>
<td>[Henderson et al., 2008]</td>
<td></td>
</tr>
<tr>
<td>VX_BR89</td>
<td>MGGA</td>
<td>[Becke and Roussel, 1989]</td>
<td></td>
</tr>
<tr>
<td>VX_MB3</td>
<td>MGGA</td>
<td>[Becke and Johnson, 2006]</td>
<td></td>
</tr>
<tr>
<td>VX_G3BJ</td>
<td>MGGA</td>
<td>[Tran et al., 2015]</td>
<td></td>
</tr>
<tr>
<td>VX_SM3BJ</td>
<td>MGGA/nonlocal</td>
<td>[Becke and Johnson, 2006]</td>
<td></td>
</tr>
<tr>
<td>VX_SLATER</td>
<td>nonlocal</td>
<td>[Slater, 1951]</td>
<td></td>
</tr>
<tr>
<td>VX_KLLI</td>
<td>nonlocal</td>
<td>[Krieger et al., 1990]</td>
<td></td>
</tr>
<tr>
<td>VX_RS</td>
<td>nonlocal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_LAPRHO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_TAU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_TAUTF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_TAUW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_TAU-TAUW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_Z</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_ALPHA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VX_ELF</td>
<td></td>
<td>[Becke and Edgecombe, 1990]</td>
<td></td>
</tr>
<tr>
<td>VX_GRR</td>
<td></td>
<td>[Tran and Blaha, 2009]</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.6: VX-switches
7.3. NLVDW

<table>
<thead>
<tr>
<th>Keyword for VC</th>
<th>Type</th>
<th>Reference</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC_NONE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VC_LDA</td>
<td>LDA</td>
<td>[Perdew and Wang, 1992]</td>
<td></td>
</tr>
<tr>
<td>VC_VWN5</td>
<td>LDA</td>
<td>[Vosko et al., 1980]</td>
<td></td>
</tr>
<tr>
<td>VC_LYP</td>
<td>GGA</td>
<td>[Lee et al., 1988]</td>
<td></td>
</tr>
<tr>
<td>VC_PW91</td>
<td>GGA</td>
<td>[Perdew et al., 1992]</td>
<td></td>
</tr>
<tr>
<td>VC_PBE</td>
<td>GGA</td>
<td>[Perdew et al., 1996]</td>
<td></td>
</tr>
<tr>
<td>VC_HCTH93</td>
<td>GGA</td>
<td>[Hamprecht et al., 1998]</td>
<td></td>
</tr>
<tr>
<td>VC_HCTH120</td>
<td>GGA</td>
<td>[Boese et al., 2000]</td>
<td></td>
</tr>
<tr>
<td>VC_HCTH147</td>
<td>GGA</td>
<td>[Boese et al., 2001]</td>
<td></td>
</tr>
<tr>
<td>VC_HCTH407</td>
<td>GGA</td>
<td>[Boese and Handy, 2001]</td>
<td></td>
</tr>
<tr>
<td>VC_AM05</td>
<td>GGA</td>
<td>[Armiento and Matsson, 2005]</td>
<td></td>
</tr>
<tr>
<td>VC_PRESOL</td>
<td>GGA</td>
<td>[Perdew et al., 2008]</td>
<td></td>
</tr>
<tr>
<td>VC_RGE2</td>
<td>GGA</td>
<td>[Ruzsinszky et al., 2009]</td>
<td></td>
</tr>
<tr>
<td>VC_PBEINT</td>
<td>GGA</td>
<td>[Fabiano et al., 2010]</td>
<td></td>
</tr>
<tr>
<td>VC_SG4</td>
<td>GGA</td>
<td>[Constantin et al., 2016]</td>
<td></td>
</tr>
<tr>
<td>VC_MPBE</td>
<td>GGA</td>
<td>[Haas et al., 2010]</td>
<td></td>
</tr>
<tr>
<td>VC_ACGGA</td>
<td>GGA</td>
<td>[Burke et al., 2014]</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.7: VC-switches

- **grad**: yes or no for the calculation of the forces on the nuclei (necessary for the minimization of internal parameters).
- **pbc**: yes or no for periodic boundary conditions (pbc). It should be no for an isolated atom or molecule in a big box.
- **abc**: yes or no for the calculation of the three-body dispersion contribution with DFT-D3.
- **cutoff <value>**: The cutoff for the dispersion interaction. The default is 95 bohr.
- **cnthr <value>**: The cutoff for the coordination number CN. The default is 40 bohr.
- **num**: yes or no for the numerical (instead of analytical) calculation of forces.

## 7.3 NL-vdW (Calculate the dispersion energy with nonlocal van der Waals functionals)

**nlvdw** calculates the dispersion energy and potential with nonlocal van der Waals (NL-vdW) functionals [Dion et al., 2004] using the FFT-based method of Román-Pérez and Soler [Román-Pérez and Soler, 2009]. Details specific to **WIEN2k** can be found in [Tran et al., 2017].

### 7.3.1 Execution

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

```
x nlvdw [-p]
```

### 7.3.2 Input

The program **nlvdw** needs the electron density on a grid in the unit cell in **case.xsf**. This file is prepared automatically by **prepare.xsf_lapw** when **run_lapw** is executed with the -nlvdw switch. The options for the **nlvdw** package have to be specified in the input file **case.innlvdw**. A template can be found in **$WIENROOT/SRC/templates/**:

```plaintext
---------------- top of file: case.innlvdw ----------------
1 kernel type
-0.8491 parameters of the kernel
20 plane-wave expansion cutoff GMAX
0.3 density cutoff rho<
T calculation of the potential (T or F)
---------------- bottom of file: --------------------
```
The options are explained below:

- **kernel type**: 1 (for the analytical form of [Dion et al., 2004]) or 2 (for the analytical form of [Vydrov and Van Voorhis, 2010, Sabatini et al., 2013]).
- **parameters of the kernel**: one value ($Z_{ab}$) for kernel type 1 (e.g., -0.8491 from [Dion et al., 2004] or -1.887 from [Lee et al., 2010]) or two values ($b$ and $C$) for kernel type 2 (e.g., $b = 6.3$ and $C = 0.0093$ for rVV10 [Vydrov and Van Voorhis, 2010, Sabatini et al., 2013]).
- **plane-wave expansion cutoff** $G_{\text{MAX}}$: 20 bohr$^{-1}$ is a relatively good value.
- **density cutoff** $\rho_{\text{cutoff}}$: 0.3 bohr$^{-3}$ is a relatively good value.
- **potential**: if set to T (true), the potential is calculated.

### 7.4 ORB (Calculate orbital dependent potentials)

This program was contributed by:

P. Novák
Inst. of Physics, Acad. Science, Prague, Czech 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 lapwmd. 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” $U_{\text{eff}} = U - J$; setting $J = 0$.
  2. LDA+U(AMF) - introduced by [Czyżyk 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...
7.4. ORB

7.4.1 Execution

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

```
x orb [ -up/-dn/-ud ] or orb up/dnorb.def
```

7.4.2 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

7.4.3 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
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

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
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’
```

nmodop

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{\hat{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
1 nmodop
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 ---------------------
7.5 LAPW1 (generates eigenvalues and eigenvectors)

lapw1 sets up the Hamiltonian and the overlap matrix ([Koelling and Arbman, 1975]) 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 [Sjostedt et al., 2000, Madsen et al., 2001]) is supported and the basis sets can be mixed for maximal efficiency. You can also set multiple LOs (low- or high-energy LOs, eg. for NMR calculations, or for including more (semi-)core states or for a better description of high-Energy states (typically 2-4 Ry above $E_F$) for spectroscopies (XANES, TELNES, optics at high energy)). When adding such LOs, make sure their $E$-parameters are far away from each other (typically at least 1 Ry, often 2.4 Ry), otherwise ghostbands may occur. In addition (or sometimes, alternatively) a second-derivative (HDLO) LO for improved $E$-linearization of valence bands (in particular for d and f-states and large RMTs) [Karsai et al., 2017] can be defined. 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 (this will be done automatically, when using the x_lapw script).

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, 1989, Blaha et al., 2009]). 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 -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. As an alternative to ScalaPack one can also use the ELPA library for diagonalization.

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.

7.5.1 Execution

The program lapw1 is executed by invoking the command:

```
x lapw1 [-c -up|dn -it -noHinv|--noHinv0 -p -nohns -orb -band
  -nmat_only -nmr] 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 `lapws1.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).

### 7.5.2 Dimensioning parameters

The following parameters (collected in file `param.inc` 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)
- **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
- **NSYM**: order of point group
- **NUME**: maximum number of energy eigenvalues per k-point
- **NVEC1**: defines the largest triple of integers which define reciprocal K-vectors when multiplied with the reciprocal Bravais matrix
- **NVEC2**: restrict output for mpi-jobs, limits the number of case.output1_X.proc_{XXX} files to “restrict_output”

### 7.5.3 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.

```plaintext
------------------ top of file: case.in1 -------------------
WFFIL EF=0.5000 (WFPRI,WFFIL,SUPWF ; wave fct. print,file,suppress
7.500 10 4 ELFA pxq hm (R-mt+K-max; MAX 1, max 1 for hns)
0 -3.00 0.020 CONT 0 ENERGY PARAMETER for s, LAPW
1 -1.90 0.020 CONT 0 ENERGY PARAMETER for p
2 0.20 0.020 CONT 1 APW
K-VECTORS FROM UNIT:4 -9.0 2.0 69 emin/emax/nband
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
------------------- bottom of file ------------------------
```

Interpretive comments follow:

**line 1**: free format

- **switch**, **EF**: standard option, writes wave functions to file `case.vector` (needed in `lapw2`)
- **SUPWF**: suppresses wave function calculation (faster for testing eigenvalues only)
- **WPRI**: 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 the default energy parameters (0.3) to “EF-0.2” or “EF+0.2” (when a “shallow-LO” is present) Ry.

line 2: free format
rkmax, lmax, lnsmmax, library, gridshape, memory

rkmax \( R_{max} \times K_{max} \) determines matrix size (convergence), where Kmax is the plane wave cut-off, Rnt is the smallest of all atomic sphere radii. Usually this value should be between 5 and 9 (APW+lo) or 6 - 10. (LAPW-basis) \( K_{max}^2 \) 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 \( R_{nt} \) (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)

lnsmmax maximum l value for partial waves used in the computation of non-muffin-tin matrix elements (lnsmax=4 is quite good, but for large spheres and highest precision one should increase it to eg. 6). Larger lnsmax may increase cputime significantly.

library ELPA ELPA is used to solve the eigenvalue problem in case of parallel calculations (default).

SCALA ScaLAPACK is used to solve the eigenvalue problem in case of parallel calculations.

Note, that this switch is only referenced, if lapw1 has been compiled with -DELPA.

gridshape pxq pxq processor grid for parallel calculations
qxp qxp processor grid (for rectangular grids; p is the large, and q the small dimension)

memory hm default for parallel calculations using ELPA. There has to be enough memory to allocate an additional array of the size of H.

lm can be used for calculations with ELPA if the memory is limited. However, a (small) performance penalty occurs due to more MPI communication.

line 3: free format
Etral, ndiff, Napw

Etral 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,I2)
I, El, de, switch, NAPWL

l l of partial wave
El \( E_l \) for \( L=1 \)
7.5. LAPW1

de energy increment
de=0: this E(l) overwrites the default energy (from line 3)
de ≠ 0: a search for a resonance energy using this increment is done.
Use very small de for semicore states or high precision total energy comparisons.
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, 2 ... use HDLO 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 more than once, local orbitals are added to the (L)APW basis, but only ONE HDLO per l-value is allowed. 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 “\( E_F - 0.2 \)”, or by “\( E_F + 0.2 \)” if a shallow semicore state (\( E_{LO} > -2.0 \text{Ry} \)) has been defined too. Please read also the comments about run_lapw in section 5.1.4. In addition, you may want to change the automatically created input and add HDLOs or d- or f-local orbitals at high energies to reduce the linearization error of valence or conduction band states, respectively (e.g. for spectroscopy you could put s, p, d, and/or f-LOs at very high energies (typically more than about 3.0 Ry) to better describe unoccupied states.
CHAPTER 7. SCF CYCLE

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

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

  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.
                        Note: When using ELPA for MPI-parallel computations EMAX does
                        NOT determine the largest eigenvalue and is ignored. To increase the
                        number of computed eigenvalues in that case you have to increase
                        nband.

  nband                number of eigenvalues calculated with iterative diagonalization and
                        when ELPA is used (for MPI-parallel computation). Set automatically
                        to nband = (ne * 2.0 + 5)/ispin (where ispin is 2 for non-spinpolarized
                        and 1 for spinpolarized cases) in lstart and init.lapw. Larger values
                        will lead to more cpu-time. (optional input)

  divide               optional keyword to trigger the divide+conquer diagonalization, which
                        could be useful for “all” eigenvalues (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 λ = 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 cartesian
                        coordinates in units of 2π/a, 2π/b, 2π/c for P, C, F and B cubic, tetragonal
                        and orthorhombic lattices, but internal coordinates for H and mono-
                        clinic/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

7.6 HF (Calculates the hybrid orbitals and eigenvalues)

hf calculates the orbitals and eigenvalues for hybrid functionals using the second-variational
procedure, i.e., the semilocal orbitals generated by lapw1 are used as basis functions for the second-
variational Hamiltonian [Tran and Blaha, 2011]. The number of these basis functions is determined by nband in case.inhf, but you have to make sure that lapw1 calculates sufficient bands, which is determined by EMAX (or in case of MPI-parallel computations with ELPA by nband) in case.in1(c). The hybrid orbitals are stored in case.vectorhf (full Brillouin zone).

Since calculations with hybrid functionals are much more expensive than with semilocal functionals, it is important to choose carefully the values of the various parameters (nband, gmax, Imaxe and Imavx) in case.inhf because the computational time will depend strongly on them. Choosing carefully the value of a parameter means to determine (by test calculations) the lowest value which is enough for the accuracy that is needed. This will depend on the solid, the property (e.g., lattice constant or band gap) and the RMT. The smaller the RMT is, the more Imaxe and Imavx can be chosen to be small, while gmax will need to be increased.

Setting up a hybrid calculation needs some additional considerations and is described in detail in Sec. 4.5.8. Parallel execution (fine grain MPI and on the k-point level) is also possible and is described in Secs. 4.5.8 and 5.5.

Beside the selfconsistent calculations, it is also possible to calculate the total energy with hybrid functionals non-selfconsistently (switch -nonself) and to calculate the hybrid eigenvalues (but not the orbitals) in a cheap way (switch -diaghf).

### 7.6.1 Execution

The program hf is executed by invoking the command:

```
x hf [-up/dn -c -p -band -diaghf -nonself -newklist -redklist
     -model1/model2] or
hf hf.def or hfc hf.def
```

### 7.6.2 Input

```
----------------- top of file: case.inhf ------------------
  0.25 alpha
T screened (T) or unscreened (F)
  0.165 lambda
xx nband
  6 gmax
  3 Imaxe
  3 Imavx
1d-3 tolu
---------------- bottom of file: ------------------------
```

Interpretive comments on this file are as follows:

**line 1:** free format

\[ \alpha \] fraction \((\alpha \in [0, 1])\) of Hartree-Fock exchange

**line 2:** free format

screening if set to F (false), no screening is applied to the exchange. If set to T (true), the exchange is screened by means of the Yukawa potential and the screening parameter \(\lambda\) will have to be specified in the next line. Note, that unscreened HF requires a denser \(k\)-mesh than screened HF.

**line 3:** free format
screening parameter in bohr$^{-1}$. This line should be present only if screening is set to T (true) in line 2. With the value $\lambda = 0.165$ bohr$^{-1}$, the results are very close to the values from the HSE06 hybrid functional [Tran and Blaha, 2011]. Values for $\lambda$ smaller than 0.0001 or larger than $\sim 5$ can eventually lead to suspicious results due to numerical instabilities.

**line 4:** free format

- **nband**
  
  the number of bands used for the 2nd variational procedure. nband should be at least equal to the number of (partially) occupied bands plus one. The choice for nband will depend strongly on the studied property and accuracy needed. If the switch `-diagf` is used, then the accuracy of the eigenvalues will not depend on the value of nband, therefore nband can be chosen as the smallest value that you want (but still at least to the number of occupied bands plus one)

**line 5:** free format

- **gmax**
  
  magnitude of the largest vector $\mathbf{G}$ in the Fourier expansion of the product of two orbitals and the generated potential in the interstitial region (Eqs. (13) and (14) in [Tran and Blaha, 2011]). gmax=6 can eventually represent a good compromise between computational time and accuracy.

**line 6:** free format

- **lmaxe**
  
  maximum value of the angular momentum for the expansion in spherical harmonics of the product of two orbitals and the generated potential inside the atomic spheres (Eqs. (13) and (14) in [Tran and Blaha, 2011]). lmaxe=3 or 4 are usually large enough for good accuracy for light elements. For systems with $f$ electrons, the value lmaxe=6 may eventually be necessary.

**line 7:** free format

- **lmaxv**
  
  maximum value of the angular momentum of the expansion of the orbitals ($\ell_i$ in Eq. (15) in [Tran and Blaha, 2011]). The value should be at least equal to the largest chemical $\ell$ present in the system.

**line 8:** free format

- **tolerance**
  
  below this value, the double radial integrals in Eq. (26) [Tran and Blaha, 2011] are neglected. With tolu=1d-3 (or even 1d-2) not much accuracy is lost.

### 7.7 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 `lapwl`. For reference see [Singh and Nordström, 2006, Novák, 1997]. 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 \texttt{lapw1} by increasing EMAX in \texttt{case.in1} (up to 10 Ry!) - if you are running MPI-parallel calculations with ELPA you have to increase nband in \texttt{case.in1} instead! We also provide an additional basis function, namely a “relativistic-LO” (RLO) with a $p_{3/2}$ 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]). It is particular helpful for heavier atoms with semicore p-states, but it must not be used for EFG calculations. 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 dependent potentials (LDA+U, ECE 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 \texttt{lapw2} should also be modified since the potential has lower symmetry than in the non-relativistic case. The following inputs may change:

\begin{itemize}
  \item \texttt{case.struct}
  \item \texttt{case.klist}
  \item \texttt{case.kgen}
  \item \texttt{case.in2c}
  \item \texttt{case.in1}
\end{itemize}

We recommend to use \texttt{initso} (see Sec.5.2.19) which helps you together with \texttt{symmetso} (see Sec.9.1) to setup spinorbit calculations.

Note: SO eigenvectors are complex and thus \texttt{lapw2c} must be used in a selfconsistent calculation.

### 7.7.1 Execution

The program \texttt{lapwso} is executed by invoking the command:

\begin{verbatim}
x lapwso [ -up -p -c -orb] or lapwso lapwso.def
\end{verbatim}

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).

### 7.7.2 Dimensioning parameters

The following parameters are used (collected in file \texttt{module.f}):

- \texttt{FLMAX} constant = 3
- \texttt{LMAX} highest l of wave function inside sphere (consistent with \texttt{lapw1})
- \texttt{LOMAX} max l for local orbital basis
- \texttt{NRAD} number of radial mesh points

### 7.7.3 Input

A sample input for lapwso is given below. It will be generated automatically by \texttt{initso}
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

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 \( E_{\text{top}} \) or \( E_{\text{bottom}} \) are not found
STOP calculation stops if not both \( E_{\text{top}} \) and \( E_{\text{bottom}} \) 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
7.8 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 -qt1). In order to get partial charges for bandstructure plots, use -band, which sets the “QTL option and uses “ROOT” in case.in2.

For meta-GGA calculations energy densities are written to case.vrepval (switch -vresp) and for DFT+U calculations the density matrices case.dmatup/dn are calculated when the switch -orb is activated.

Some other switches change the input file case.in2 temporarily and are described in the input section.

7.8.1 Execution

The program lapw2 is executed by invoking the command:

```
x lapw2 [-c -up|dn -p -so -orb -qt1 -fermi -efg -hf -redklist
-band -eece -vresp -help_files -emin X -all X Y -scratch dir
-alm -almd -qdmft -in1orig] or
lapw2 lapw2.def [proc#] or lapw2c lapw2c.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.

7.8.2 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 <= 8)
- **LOMAX** max l for local orbital basis
- **NCOM** number of LM terms in density
- **NGAU** max. number of Gaunt numbers
CHAPTER 7. SCF CYCLE

NRAD number of radial mesh points
restrict_output for mpi-jobs, limits the number of case.output2_X_proc XXX files to “restrict_output”

7.8.3 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 1 (EMIN, # of electrons,ESEPERMIN, ESEPER0,iqtlsave)
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
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. 5.1.4)
switch FOR partial charges only (generates file case.qtl for DOS calculations), set
automatically by switch -qtl
switch QTL computes decomposition of electric field gradient (EFG), contributions
from inside spheres (the total EFG is computed in lapw0), set automat-
ically by switch -efg.
switch EFG this generates two files, case.radwf and case.almblm, where the
radial wavefunctions and the A_{lm}, B_{lm}, C_{lm} coefficients of the wave-
function inside spheres are listed. Do not set it manually, but using the
-almd switch. The file case.almblm can get very big.
switch 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 wavefunc-
tion inside spheres are listed in a format usable for the TRIQS DMFT
code. Do not set it manually, but using the -almd switch. The file
case.almblm can get very big.
switch ALMD this generates two files, case.radwf and case.almblm, where the ra-
dial wavefunctions and the A_{lm}, B_{lm}, C_{lm} coefficients of the wavefunc-
tion inside spheres are listed in a format usable for the TRIQS DMFT
package, set automatically by switch -fermi.
switch CLM CLM charge density coefficients only
switch FERMI Fermi energy only, this produces weight files for parallel execution
and for the optics and lapwdm package, set automatically by switch
-femi.
switch EECE if set to “EECE”, calculates the density for specified atoms and angu-
lar momentum only. Used for exact-exchange or hybrid-calculations,
usually set automatically by runsp_lapw -eece

line 2: free format
emin, ne, esepermin, eseper0, iqtlsave
emin  lower energy cut-off for defining the range of occupied states, can be
set temporarily 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 \((EF - \text{esepermin})\) 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
iqtlsave  optional value, checks if the low-energy band ranges (below -2 Ry) are
“narrow” (below 0.2 Ry) and stops (iqtlsave=1 = default) / does not
stop (iqtlsave=0). You may have to switch it off for extreme pressures,
because then you may have large band width even for semi-core states.

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 \( T=0 \)K. (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 conver-
gen, 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 et al., 1994]. This “standard” scheme is recommended for
optic. In this case the file case.kgen, consistent with the \( k \)-mesh
used in \text{lapw1}, must be provided (see Sec. 7.5). This is the recom-
manded option although convergence may be slower than with Gauss-
or temperature-smearing. TETRA may lead to problems (wrong num-
ber of electrons) for 2D-BZ meshes (as for surface slabs or cells with
a large \( c \)-parameter) and a mesh-division of “1”. Use TEMP in these
cases.
ALL  All states up to eval are used. This can be used to generate charge den-
sities in a specified energy interval, can be set temporarily 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 temporarily by switch \texttt{--all X Y}), if efmod is set to TETRA, eval \texttt{.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 $D = 1.0 \times 10^{-6}$. You can modify this by setting eval equal to your desired $D$ (or $100+D$).

\textbf{optional line 3a:} free format (ONLY when EECE is set)

\textbf{nat\_rho} number of atoms for which a specific density should be calculated

\textbf{optional line 3b:} free format (ONLY when EECE is set)

\textbf{jatom\_rho, L\_rho}

\textbf{jatom\_rho} index of atom for which a specific density should be calculated

\textbf{L\_rho} angular momentum l-value for which a specific density should be calculated

\textbf{line 4:} format (121(I3,I2))

\begin{verbatim}
L,M LM values of lattice harmonics expansion (equ. 2.10), defined according
to the point symmetry of the corresponding atom; generated in SYM-
METRY, MUST be consistent with the local rotation matrix defined in
case.conv (details can be found in [Kara and Kurki-Suonio, 1981]).
CAUTION: additional LM terms which do not belong to the lattice har-
onics will in general not vanish and thus such terms must be omitted.
Automatic termination of the \textit{lm} series occurs when a second 0,0 pair
appears within the list. When you change the \textit{l, m} list during an SCF
calculation the Broyden-Mixing is restarted in \texttt{mixer}.
\end{verbatim}

\textbf{line 5:} free format

\textbf{GMAX} max. G (magnitude of largest vector) in charge density Fourier expan-
sion. 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 deter-
mined by GMAX. When you change GMAX during an scf calculation
the Broyden-Mixing is restarted in \texttt{mixer}.
Table 7.55: LM combination and local coordinate system of “non-cubic groups” (requires “negative atomic index” in case.struct)

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

7.9 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.

7.9.1 Execution

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

```
x sumpara -d [-up/-dn/-ud -hf] 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.

7.9.2 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
7.10 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:

\[ X = X_r(r) \ast X_{ls}(\vec{l}, \vec{s}) \]

\( X_r(r) \) 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 \( \frac{1}{r^3} \) expectation value inside the atomic sphere (must be multiplied by g=2 to get the spin moment)
- RINDEX=1 LSINDEX=2: <X> is the projection of the electronic spin inside the atomic sphere (in case of SO-calculations WITHOUT LDA+U)
- RINDEX=1 LSINDEX=3: <X> is the projection of the orbital moment inside the atomic sphere (for a converged calculation at the very end)
- 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 [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.

7.10.1 Execution

The program lapwdm is executed by invoking the command:

\[ x \text{ lapwdm } [-up/dn -p -c -so -hf] \text{ or } \text{lapwdm lapwdm.def} \]

7.10.2 Dimensioning parameters

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

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

7.10.3 Input

A sample input for lapwdm is given below.

------------------ top of file: case.indm --------------------
-9. Emin cutoff energy
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, RMTRED
   RINDEX 0-3, as described in the introduction to lapwdm
   LSINDEX 0-5, as described in the introduction to lapwdm
   RMTREDN, optionally allows to check RMT-reduction (by N points) of expectation values

7.11  LCORE (generates core states)

lcore is a modified version of the [Desclaux, 1969, Desclaux, 1975] 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.
7.11.1 Execution

The program \texttt{lcore} is executed by invoking the command:

\begin{verbatim}
lcore lcore.def or x lcore [-up|-dn]
\end{verbatim}

For the spin-polarized case see fcc Ni on the distribution tape. If \texttt{case.incup} and \texttt{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.

7.11.2 Dimensioning parameters

The following parameter is listed in file \texttt{param.inc}:

\begin{verbatim}
NRAD number of radial mesh points
\end{verbatim}

7.11.3 Input

Below is a sample input (written automatically by \texttt{lstart}) for TiO$_2$ (rutile), one of the test cases provided with the \texttt{WIEN2k} package.

In case of a "open core" calculation (e.g. for 4f states) you may need "spin-polarized" \texttt{case.inc} files in order to define different configurations for spin-up and dn. Create two files \texttt{case.incup} and \texttt{case.incdn} with the corresponding occupations. The \texttt{runsp_lapw} script will automatically copy the corresponding files to \texttt{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 zs
2,-2,4 zp
2, 1,2 zp*
1 0.0 # of orbital of second atom
1,-1,2 ls
$ end switch
--- bottom of file ---
```

Interpretive comments on this file are as follows:

- **line 1**: free format
  - \texttt{norb}, \texttt{shift}, \texttt{iwarn}

  \begin{verbatim}
norb shift iwarn
  \end{verbatim}

  - \texttt{norb} number of core orbitals
  - \texttt{shift} shift of potential for "positive" eigenvalues (e.g. for 4f states as core states in lanthanides)
  - \texttt{iwarn} optional print switch to reduce (0) or increase (1) printing to \texttt{case.outputc}

- **line 2**: free format
  - \texttt{n}, \texttt{kappa}, \texttt{occup}

  \begin{verbatim}
n kappa occup
  \end{verbatim}

  - \texttt{n} principle quantum number
  - \texttt{kappa} relativistic quantum number (see Table 6.6)
  - \texttt{occup} occupation number (including spin), fractional occupations supported
7.12 MIXER

>>> line 2 is repeated for each orbital (norb 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

7.12 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

\[ \rho_{\text{new}}(r) = (1 - Q)\rho_{\text{old}}(r) + Q\rho_{\text{output}}(r) \]

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, 2013]) 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 onsite-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.

7.12.1 Execution

The program mixer is executed by invoking the command:
mixer mixer.def or x mixer [-eece -dftd3]

A spin-polarized case will be detected automatically by x due to the presence of a case.clmvalup file. For an example see fcc Ni (sec. 10.2) in the WIEN2k package.

7.12.2 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

7.12.3 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 --------------------
MSR1 0.00 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
## VLSOW, SLOW, FAST
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

**line 1: (A5,*)**
- switch, bgch, norm

  - MSR1, MSR2, MSR3: Similar to MSR1, but with updated scaling, regularization and other improvements.
  - MSEC1: Similar to MSEC1, but with improved scaling.
  - MSEC2: Similar to MSEC1, but with improved scaling.
  - MSEC3: Similar to MSEC1, but with improved scaling.
  - MSEC4: Similar to MSEC1, but with improved scaling.

  - bgch: Background charge to apply to the cell (e.g. use +1 if the system contains an additional electron or -1 to screen a core hole if it is not neutralized by an additional valence electron)
7.12. MIXER

| 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, less important for other methods. In these methods in the first iterations the default Q (0.2) is automatically adjusted and reduced/increased by the program depending on the average charge distance :DIS and the relative improvement in the last cycle. In case of too large charge fluctuations (divergence and stop of the scf cycle), Q can be reduced but this can lead to stagnation. In particular Q=0.1 switches to a “slow” mode. One should rarely reduce this below 0.1.

**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)

```
broyd, nuse
```

```
broyd
```

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 the default).

**line 5 (optional line):** (free format)

```
trust
```

```
VSLOW
```

For very difficult cases, where divergence (like spin-polarized systems or with many TM atoms) or endless oscillations occur.

```
SLOW
```

```
FAST
```

For easy cases to accelerate (also MSR1a).

In addition, mixer reads a “mixing factor” from file .pratt or .msec, which can be used during scf/MSR1a optimizations or at the very beginning to push convergence. You can create it using

```
echo 0.2 > .pratt
```

These files will be removed automatically once they are used. For additional documentation consult the README file in SRC.mixer.
8 Programs for analysis, calculation of properties, and geometry optimization

Contents

8.1 AIM ................................................................. 145
8.2 BerryPI ......................................................... 149
8.3 BROADENING ................................................. 150
8.4 DIPAN ............................................................... 151
8.5 ELAST .............................................................. 153
8.6 FILTVEC .......................................................... 157
8.7 FSGEN .............................................................. 159
8.8 IRelast .............................................................. 159
8.9 IRREP ............................................................... 161
8.10 JOINT .............................................................. 161
8.11 KRAM .............................................................. 164
8.12 LAPW3 ........................................................... 166
8.13 LAPW5 ........................................................... 166
8.14 LAPW7 ........................................................... 169
8.15 MINI ............................................................... 172
8.16 NMR ............................................................... 175
8.17 OPTIC ............................................................. 175
8.18 OPTIMIZE ....................................................... 179
8.19 QTL ............................................................... 180
8.20 SPAGHETTI ...................................................... 182
8.21 TELNES3 ......................................................... 185
8.22 TETRA ............................................................ 192
8.23 XSPEC ........................................................... 195

8.1 AIM (atoms in molecules)

This program was contributed by:
This program analyses the topology of the electron density according to Bader’s “Atoms in molecules” theory. For more information see [Bader, ] 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 $start$, sect. 6.4), otherwise discontinuities appear at the sphere boundary.

### 8.1.1 Execution

The program `aim` is executed by invoking the command:

```
aim aim.def or aimc aim.def or x aim [-c ]
```

### 8.1.2 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

### 8.1.3 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/THREE/ALL /FOUR
3 3 3 # x,y,z nshells (of unit cells)
END
----------------- bottom of file ---------------------
```

Interpretive comments on this file are as follows:
8.1. AIM

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 5.2.12) and get in the file critical_points_ang a comprehensive list of the CP (sorted and unique) with all values given in Å, e/Å³, ... (instead of bohr).

----------------- top of file: case.inaim ------------------

SURF atom in center of surface (including MULT)
3 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 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

```fortran
  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

```fortran
  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=\pi` 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

```fortran
  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

```fortran
  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

```fortran
  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

```fortran
  nxsh, nysh, nzsh
```

specifies the number of nearest neighbor cells (in x,y,z direction) where atomic positions are generated.

**line 8:** A4

```fortran
  IRHO
```

integrate function on “unit 9” (usually `case.clmsum`) inside previously defined surface (stored in `case.surf`).
8.2. BERRYPI

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.

### 8.2 BerryPI (Modern theory of polarization)

This program was contributed by:

| S.J. Ahmed, J. Kivinen, B. Zaporzan, L. Curiel, S. Pichardo, O. Rubel |
| Thunder Bay Regional Research Institute, Ontario, Canada |
| Computer Physics Communications 184, 647-651 (2013) |
| Sources available from: [https://github.com/spichardo/BerryPI](https://github.com/spichardo/BerryPI) |
| Email: rubelo@tbh.net |

This program calculates the spontaneous polarization, Born effective charges or piezoelectric constants using the Berry phase approach. More details about its usage are given in Chapter 5.8.

It consists of a set of Python scripts (requires Python 2.7 and the NumPi library) and uses **wien2wannier** for the calculation of overlap integrals. The main steps of a ‘‘**berrypi -kNX:NY:NZ**’’ call include:

- x kgen -fbz
  - generate a k-mesh in the full Brillouin zone
- x write_inwf
  - Prepare the input for w2w with the occupied band range
- x write_win case
  - Create the input file for w2w
- x win2nnkp.py case
  - Generate the nearest neighbor list of k-points
- x lapw1
  - Calculate wavefunctions for the new k-list
- x w2w
  - Calculate the overlap matrix $S_{mn}(k_j, k_{j+1})$
- x lapwso and x w2waddsp (only in case of SO)
- x mmn2pathphase.py case x
  - Calculate the Berry phase along x-axis
8.3 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 (broadening of case.elnes) or the xspec program (broadening of case.xspec) to broaden theoretical spectra by applying a lorentzian broadening for core and valence life times and a gaussian broadening for spectrometer broadening. The output file will be called case.broadspec.

8.3.1 Execution

Execution

The program broadening is executed by invoking the command:

```
broadening broadening.def or x broadening [-xspec -up/-dn]
```

8.3.2 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
```
8.4 DIPAN

This program was contributed by:

P. Novák
Inst. of Physics, Acad. Science, Prague, Czech 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.

The magnetic field is given by

$$B = \frac{\mu_0 \mu}{4\pi r^3} \left[ 3(\hat{n} \cdot \hat{r}) \hat{r} - \hat{n} \right] \tag{8.1}$$

where \(\hat{r} = \vec{r}/r\).
\(\hat{n} = \vec{M}/M\) is direction of magnetization.
\(\mu_0\) is permeability of free space; \(\mu_0 = 4\pi \times 10^{-7}\) 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 \(\mu\) in Bohr magnetons \(\mu_B = 9.274078 \times 10^{-24}\) J/T and \(r\) in atomic units for length \(a_0\) (Bohr radius) \(a_0 = 5.2917706.10^{-11}\) m.

Inserting in (1) gives

$$B = 6.258463 \frac{\mu(\mu_B)}{r(\text{a.u.})^3} \left[ 3(\hat{n} \cdot \hat{r}) \hat{r} - \hat{n} \right]. \tag{8.2}$$
Total dipolar field acting on atom \(i\) is given by the lattice sum

\[
\vec{B}_i = 6.258463 \sum_j \frac{\mu_j}{r_j^3} [3(\vec{n}\hat{r}_j)\hat{r}_j - \vec{n}].
\]

(8.3)

Dipolar anisotropy energy is given by the sum

\[
E_{an} = -\frac{1}{2V} \sum_j \vec{B}_j \mu_j
\]

(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 \(T\), \(\mu_j\) in \(\mu_B\) and \(V\) in \((\text{a.u.})^3\) gives

\[
E_{an}(J/m^3) = -\frac{3.129232 \times 10^7}{V(\text{a.u.})^3} \sum_j \vec{B}_j(T)\mu_j(\mu_B)
\]

(8.5)

### 8.4.1 Execution

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

```
dipan dipan.def or x dipan
```

### 8.4.2 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

### 8.4.3 Input

An example is given below:

```
---------------- top of file: case.indipan -----------------------
160. 0 Rmax (a.u.), ipr (printing option)
-0.26 Magnetic moment of 1st 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: number 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**
8.5. ELAST

line 2: free format

mm Magnetic moment ($\mu_B$) 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

8.5 ELAST (Elastic constants for cubic cases)

This program was contributed by:

original author: 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).

modified by Ferenc Karsai
Institute for MaterialsChemistry
TU Vienna
ferenc@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 calculates elastic constants for cubic crystals. It is described in detail by the author in
[Charpin, 2001].

8.5.1 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 rhombo-
  hedral 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.

The following modifications of init_elast, elast_setup and ana_elast prepare input files for calculations of elastic constants at different pressures and analyze the results:

init_elast_pressure:
As in the case of init_elast the script is called in the working directory with a valid case.struct file and requires an input file case.inelastp1 (a template can be found at $WIENROOT/SRC_templates/template.inelastp1). The script creates the directory elast/ with the necessary subdirectories pressure_x/ according to the number x of pressure changes in the case.inelastp1 input file and the templates for isotropic, tetragonal and rhombohedral (trigonal) distortions at each pressure (pressure_1/eos, pressure_1/tetra, pressure_1/rhomb, pressure_2/eos, ...). In each pressure_x/ directory a file called z_pressure.dat is created with the lattice constant at each pressure given in case.inelastp1. In contrast to init_elast the calculations are initialized using init_lapw in batch mode and the necessary parameters are set in case.inelastp1. The following small programs and scripts are utilized by init_elast_pressure: iniel_pressure_reader.pl, iniel_pressure_in2reader.pl, genetempl

elast_setup_pressure: Similar to elast_setup this script has to be run in the elast directory and requires the input file elast.inelastp2 (a template can be found at $WIENROOT/SRC_templates/elast.inelastp2). It creates the distorted struct files and the pressure_x/eos.job, pressure_x/rhomb.job, pressure_x/tetra.job and pressure_x/runjob.x files in each directory pressure_x. The three scripts eos.job, rhomb.job and tetra.job can either be started separately or together by runjob.x. The number of structure changes per pressure and the calculational parameters are set in elast.inelastp2. The following small programs and scripts are utilized by elast_setup_pressure: elast_setup_input.pl, setelast_pressure

ana_elast_pressure: Once all calculations are done, this script (in contrast to ana_elast) has to be run in the elast directory. It requires the pressure_x/z_pressure.dat files created by init_elast_pressure. The final results for a given pressure are stored in pressure_x/elast/result/outputs. Additionally the collective results for all pressure are stored in the directory elast_results. If the script is called with the option --plot (eg. ana_elast_pressure --plot) then postscript files for the fits using gnuplot are created in the pressure_x/results/outputs directory. The following small programs and scripts are utilized by ana_elast_pressure: anaelast_pressure

8.5.2 Input

Below are examples for case.inelastp1 and elast.inelastp2:

---------------- top of file: case.inelastp1 -------------------
15 RMT_reduction
XC_PBE V_xc_potential
-6 CORE_separation
9 RKMAX
Interpretive comments on this file are as follows:

**line 1**: free format

RMT

1-100  RMT reduction by X %
OLD  RMT values taken from case.struct file in working directory

**line 2**: free format

\( V_{xc} \)  Exchange-correlation potential

**line 3**: free format

\( E_{sep} \)  Energy separation for core and valence states

**line 4**: free format

RKMAX  RKMAX value

**line 5**: free format

k-mesh  Number of k-points in full BZ

**line 6**: free format

GMAX  GMAX value

**line 7**: free format

SPIN   NM  non magnetic
SPIN   SPIN  spin polarized

**line 8**: free format

INST   OLD  case.inst file is taken from working directory
NEW  new case.inst file is created

**line 9**: Empty line

**line 10-x**: free format

\( a, p \)

\( a \)  lattice constant in a.u. at pressure \( p \) (determined from e.g. a previous volume optimization ...)

\( p \)  pressure \( p \) written in \texttt{pressure.x/z/pressure.dat}
The `elast.inelastp2` file looks like:

```
---------------- top of file: elast.inelastp2 -------------------
iso 0
tet 0
trig 5
-2
-1
0
1
2
ec 0.0001
spin .FALSE.
parallel .TRUE.
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

**line 1-3 (optional):** free format
distortion, n

- distortion: if this line is given then the specified distortions will be calculated
- iso: isotropic distortion
- tet: tetragonal distortion
- trig: trigonal (rhombohedral) distortion

- n: number of structure changes n for a given type of distortion; the exact changes in the lattice constant are given on the following n lines (free format, lines 4-8 in the example above)

- 0: default values are taken (−10%, −9%, ..., −1%, 0%, 1%, ..., 9%, 10% → 21 values)

- > 0: change in the lattice constant in %

**line 9 (optional):** free format

- ec: energy convergence criterion (if this line is missing then default value of 0.00001 is used)

**line 10 (optional):** free format

- spin .FALSE.: no spin polarization (default)

- .TRUE.: spin polarization (`runsp_lapw` used instead of `run_lapw` in `eos.job`, `tetra.job` and `rhomb.job`)

**line 11 (optional):** free format

- parallel .FALSE.: default

- .TRUE.: if `.machines` exists in the `elast/` directory then it will be copied into `pressure.x/eos, pressure.x/tetra, pressure.x/rhomb` directories
8.6 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. While the structure of the generated **case.vectorf** file is identical to that of the original **case.vector** file, the corresponding **case.energy** file is not updated. Hence, **case.vectorf** can be used as substitutes for **case.vector** only in **lapw7**. 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**.

8.6.1 Execution

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

```
  filtvec filtvec.def  or  filtvecc filtvec.def  or  x filtvec [-c] [-up|dn] [-hf]
```

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.

8.6.2 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**.
CHAPTER 8. ANALYSIS, PROPERTIES AND OPTIMIZATION

8.6.3 Input file case.inf

Two examples are given below. The first uses global selection mode; the second individual selection mode.

I. Global Selection Mode

```
158
3 1 17 33  # number of k-point list items, k-points
2 11 -18  # number of bands, band indices

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

```
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

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.
8.7 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 then 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_en` (line 10, see sec. 8.20)
- Edit `case.spaghetti_en` 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_en`)
  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_en` 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.)

8.8 IRelast (Elastic constants for cubic, hexagonal, tetragonal, orthorhombic, monoclinic and rhombohedral cases)

This program was contributed by:

| Author: Morteza Jamal |
| Ghods City-Tehran, Iran |
| 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 ([Jamal et al., 2016]) calculates elastic constants for cubic, hexagonal, orthorhombic, tetragonal, monoclinic and rhombohedral symmetry, respectively. It replaces previous versions on our "unsupported_software" page.
CHAPTER 8. ANALYSIS, PROPERTIES AND OPTIMIZATION

The package is driven by the following scripts:

▶ set_elast_lapw:
   It prepares the whole calculation and should be run in a directory with a valid case.struct file. It finds the symmetry (S= C(cubic), H(hexagonal), T(tetragonal), O(orthorhombic), M(monoclinic) or R(rhombohedral) of the structure defined in case.struct and calls S_set_elast_lapw. It creates the necessary subdirectories elast-constant, elast-constant/c11, elast-constant/c22, ... and copies information of the present working directory into those new directories. command_init_lapw gets information to produce auto_init_lapw for automatic initialization. Then it gets the options for running the scf-cycle in the job files using S_command_run_lapw. Finally, it generates the distorted struct-files and symm.job files, where symm stands for CUBIC, HEX, TETRA, ORTHO, MONO, and RHOM, using the S_setupc11, S_setupc22, ... programs.

▶ modifyjob_lapw:
   allows you to edit and modify the previously created symm.job files. This step is not necessary when you have specified proper commandline options previously.

▶ calljob_lapw:
   will execute all produced job files in elast-constant/c11/case, elast-constant/c22/case, ... sequentially, but eventually you may run all those jobs by yourself on different machines in parallel, as these steps can take quite some time. Once all calculations are done:

▶ call_elast_lapw:
   calls all S_ana_elast_lapw and S_ana_elastic_lapw scripts and determines the elastic constants C_{ij} as well as the Voigt, Reuss, Hill, Bulk, Shear and Young modulus and the Poisson ratio (using data from the auxiliary program S_InverseELC). Using data from the auxiliary programs S_InverseELC and MassRho, S_ana_elast_lapw calculate the Sound Velocity and the Debye temperature. The main output files are case.output_elastic and INVELC-matrix, which contains the elastic compliance constants (invers of the elastic constants matrix). Finally, S_ana_elastorder_lapw will check the sensitivity of the results to the order of the polynomial fit (stored in file output-order) and, for monoclinic crystals, the TWS program which transforms the elastic constants from WIEN2k to STANDARD Cartesian coordinates (in file STDELC-matrix).

When you know your symmetry, you can simply call the corresponding series of scripts (S = C, H, T, O, M or R):

▶ S_set_elast_lapw
▶ S_modifyjob_lapw
▶ S_calljob_lapw
▶ S_ana_elast_lapw

After a first run you may check your results using more datapoints (more or different displacements). This can be done conveniently by S_setupcXX, where XX = 11, 12, ..., which should be run in the corresponding elast-constant/cXX/case directory. When you specify in addition to new datapoints also your “old” displacements, these old results will be automatically taken into account in the analysis without recalculating them.

On the other hand, when you want to change some computational parameters (RKmax, k-mesh, XC-potential) you can call command_init_lapw after S_setupcXX and then modify your symm.job file specifying “set answscf=no” and a modified “savename” (eg. pbe_rkm8).

After these preparations, you can rerun symm.job and S_ana_elast_lapw and check if your elastic constants are converged with respect to computational parameters.

Additional information can be found in $WIENROOT/SRC_IRRelast/guide.
8.9 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 useful 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.

8.9.1 Execution

The program irrep is executed by invoking the command:

```
irrep [up/dn]irrep.def or x irrep [-so -up/dn -hf]
```

8.9.2 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

8.10 JOINT (Joint Density of States)

This program was contributed by:
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 ($\varepsilon_2$) 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 $\varepsilon_2$ 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 column is the energy mesh, the second and third columns 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 referred to the higher and lower energy core state.

### 8.10.1 Execution

The program `joint` is executed by invoking the command:

```
joint joint.def or x joint [-up|dn] [-hf]
```

### 8.10.2 Dimensioning parameters

The following parameter is listend in files `param.inc`:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSYM</td>
<td>order of point group</td>
</tr>
<tr>
<td>MG0</td>
<td>number of columns (usually 9)</td>
</tr>
</tbody>
</table>

### 8.10.3 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
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
```
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
- units 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
- switch 1 joint DOS as sum over all band combinations
- switch 2 DOS for each band
- switch 3 DOS as sum over all bands
- switch 4 imaginary part of the dielectric tensor ($\epsilon_2$)
- switch 5 imaginary part of the dielectric tensor for each band combination
- switch 6 intraband contributions: number of “free” electrons per unit cell assuming bare electron mass (calculated around $E_F \pm 10 \times de$ as defined in input line 4), plasma-frequency
in addition to switch 6 the contributions from different bands to the plasma frequency are analyzed.

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!

8.11 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/(Ωcm)) 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 (without spin-orbit), the Kramers-Kronig analysis is NOT really additive, i.e. most quantities (like ϵ₁) 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 ϵ₂ (in case.jointup and case.jointdn) using addjoint-updn_lapw (this will produce case.joint) before calling (non-spinpolarized) x kram. (For a metal, the Plasma-frequencies (intraband transitions) for up and dn should be added, but then divided by √2, before using 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.
8.11. KRAM

8.11.1 Execution

The program kram is executed by invoking the command:

```
kram kram.def or x kram [-up|dn]
```

8.11.2 Dimensioning parameters

The following parameters are listed in files `param.inc`:

- **MAXDE**: maximum number of points in energy mesh
- **MPOL**: fixed at 6

8.11.3 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**: Intraband contributions are not added
  - **0**: 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)
8.12 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.

8.12.1 Execution

The program lapw3 is executed by invoking the command:

```
lapw3 lapw3.def or lapw3c lapw3.def or x lapw3 [-c ]
```

8.12.2 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

8.13 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)

In order to get 3D-data for plotting with xcrysden, you can also use the script prepare xsf lapw (see Sect. 5.10.10).

A recent extension by L.D. Marks allows to calculate STM images (constant current) according to the Tersoff-Hamman approximation. Before doing this, you have to run lapw2 with a suitable energy window around the Fermi energy, which should correspond to the experimental bias voltage (x lapw2 -all EMIN EMAX). The output contains the z-position (height) of the "tip", i.e. the position where the density has the specified value.

It is strongly recommended that you use "Run Programs Tasks Electron density plots" from w2web, see the TiC example in Fig.3.6.
8.13.1 Execution

The program lapw5 is executed by invoking the command:

```
lapw5 lapw5.def or lapw5c lapw5.def or x lapw5 [-c -up|dn]
```

8.13.2 Dimensioning parameters

The following parameters are listed in file \texttt{param.inc}:

- \texttt{LMAX2} highest \textit{L} in in LM expansion of charge and potential
- \texttt{NCOM} number of LM terms in density
- \texttt{NRAD} number of radial mesh points
- \texttt{NPT00} number of radial mesh points beyond RMT
- \texttt{NSYM} order of point group

8.13.3 Input

An example is given below. You may want to use XCRYSDEN by T.Kokalj to generate this file (see sect. 9.29.2).

```
---------------- 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/NONORTHQ
GAUSS   # this and the following lines are for STM mode
0.5 0.5 0.0 0.0 0.05 # vibrational tensor
STM 4.0D-5 3 # STM mode, density-level, axis (3=z-axis)
SEMPER # optional output format for semper7 code
FAST # optional, useful for a first crude check
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

line 1: free format

\texttt{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 \textit{x}=\textit{ix}/idv, ... in fractional units of the lattice vectors (except \textit{fc}, \textit{bc} and \textit{c} lattices, where the lattice vectors of the conventional cell are used). Note the special meaning for STM mode described below.

line 2: free format

\texttt{ix, iy, iz, idv} coordinates of x-end

line 3: free format
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. The most convenient way to specify this plane (for a more complex structure) is using XCrysDen, where you can simply click on 3 atoms which will span the plane.

**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 (optional):** 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).

**line 9 (this and the following lines are optional for the STM mode):** free format
VIBR    GAUSS  suggested mode of vibrational smearing

line 10: free format
V11,V12,V22,V13,V23,V33
the size matrix components of the vibration tensor in units of distance
squared, where the units are either ANG or ATU as defined earlier

line 11: free format
MODE, level, axis

MODE    STM  enables STM mode
level    the density level (typically $10^4 - 10^5 e^-/ang^3$). If this value is inap-
          propriately chosen, the code will terminate with a statement: “Cannot
          Bracket, sorry”.
axis     the axis normal to the surface (e.g. 3 for z-axis). Note that in STM
          mode the z-coordinate specified in the first 3 lines is used as a starting
          value for the search of the z-position where the density has the value
          of “level”. This starting z-value has to be in the interstitial (vacuum)
          region.

line 12 (optional): free format

SEMPER  this keyword puts the output in a format readable by the semper7 code
         (exchange of x,y order).

line 13 (optional): free format

FAST    this keyword performs a fast approximate calculation for checking if
         your input (in particular the density level) is reasonable.

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.

8.14 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, its 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.

### 8.14.1 Execution

The program lapw7 is executed by invoking the command:

```
lapw7 lapw7.def or lapw7c lapw7.def or x lapw7 [-c] [-up|dn] [-sel]
[-hf]
```

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.

### 8.14.2 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.
8.14. LAPW7

8.14.3 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(THOGONAL) | 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|AU LARGE|SMALL
1 0        # k-point, band index
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows.

**line 1:** 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. \( n = 0, 1, 2, \text{ or } 3 \).
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 \langle \text{blank} \rangle 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     ... In case of an n-dim. grid, first the number of grid points along each axis,
npo    ... 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, \( 2 \times n \) integers must be provided; for arbitrary lists of grid points two intergers are expected.
```

**line 5:** format(A3)
```
tool DEP     post-processing of the wave functions
           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.
```
CHAPTER 8. ANALYSIS, PROPERTIES AND OPTIMIZATION

**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 À 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 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.
  or STOR(E) wave function is selected in the previous input line.
  READ Previously stored augmentation coefficients are read in (from or RE- 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.
  or anything Augmentation coefficients are generated from the wave function information in case.vector(\( f \)).

**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 [Gay, 1983],...
8.15. MINI

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/template.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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXIT</td>
<td>maximum number of geometry steps</td>
</tr>
<tr>
<td>NRAD</td>
<td>number of radial mesh points</td>
</tr>
<tr>
<td>NCOM</td>
<td>number of LM terms in density</td>
</tr>
<tr>
<td>NNN</td>
<td>number of neighboring atoms for nn</td>
</tr>
<tr>
<td>NSYM</td>
<td>order of pointgroup</td>
</tr>
</tbody>
</table>

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 PORT Modus of the calculation
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
\[
R^{\tau+1}_m = R^\tau_m + \eta_m (R^\tau_m - R^{\tau-1}_m) + \delta_m F^\tau_m
\]
where \(R^\tau_m\) and \(F^\tau_m\) are the coordinate and force at time step \(\tau\). When the force has changed its direction from the last to the present timestep (or is within the tolerance TOLF), \(\eta_m\) 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: mbc.inM -------------------
NOSE  (NOSE/MOLD (a4))
  58.9332  400.  1273.  5.0  (Masse, delta t, T, nose-frequency)
------------------- 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
8.16 NMR (chemical shielding)

This program calculates the orbital contribution to the NMR (chemical) shielding (the total shielding for insulators). It will first calculate the perturbation of the wave functions due to the magnetic field (first order perturbation theory) and the resulting current. This induced current is then integrated (via the Biot-Savart law) to obtain the magnetic shielding at a nucleus. For details see [Laskowski and Blaha, 2012a, Laskowski and Blaha, 2012b, Laskowski et al., 2013, Laskowski and Blaha, 2014].

The program does not need to be called by the user, but it is interfaced with the script `x_nmr_lapw` (all details can be found in sect. 5.6), where the different modes/options can be selected as switches. It can run in k-point as well as in mpi-parallel mode.

It does not have its own input file, but a modified `case.in1` is necessary, which needs to be generated by `x_nmr_lapw -mode in1`. We need an extended basis set with several local orbitals (up to very high energies) for all ""l + 1"" states, where ""l"" refers to the maximal ""chemical l"" of the specific atom (l=1 for C, but 2 for Fe, ..). In addition ALL eigenvalues must be calculated, which increases the cpu-time of lapw1 as compared to a normal scf-calculation. In addition `lapw1/2` and `nmr` is run for 7 different k-meshes, an unshifted one as well as plus/minus shifted meshes in x, y and z direction.

8.17 OPTIC (calculating optical properties)

This program was contributed by:
The theoretical background is described in detail in [Abt et al., 1994] and [Ambrosch-Draxl and Sofo, 2006] (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

\[ M_i = \langle n' \vec{E}_i | \vec{p} | n \vec{E}_i \rangle^2 \]

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., 2012] 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 purpose, the following procedure should be followed:

1. copy case.struct to case.ksym (cp case.struct case.ksym);
2. generate a k-mesh in the whole Brilouin zone (x kgen -so -fbz);
3. change TOT to FERMI in case.in2c;
4. set IPRINT=1 in case.inc to activate core-wavefunction output;
5. for metallic systems, put TETRA with value 101;
6. execute x lapw1 -up / -dn;
8.17. OPTIC

- execute x lapwso -up;
- execute x lapw2 -fermi -up / -dn;
- execute x lcore -up / -dn;
- run optic: x optic -so -up;
- run joint: x joint -up.

You must not use p-1/2 “relativistic” LOs in LAPWSO and no HDLOs in lapw1, since this basis is not supported on OPTICS yet. However, you can use multiple LOs (high-energy LOs) provided you do not use the XMCD option.

8.17.1 Execution

The program optic is executed by invoking the command:

```
optic(c) optic.def or x optic [-c -up|dn -so -p -hf]
```

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.

8.17.2 Dimensioning parameters

The following dimensioning parameters (listed in param.inc.r and param.inc.c) are used:
CHAPTER 8. ANALYSIS, PROPERTIES AND OPTIMIZATION

LMAX  highest $l+1$ in basis function inside sphere (reducing LMAX to 4 or 5 may dra-
matically 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

8.17.3 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, maximal number of k-points, number of k-point to start calculation
nkfirst

**line 2:** free format
emin, absolute energy range (Ry) for which matrix elements should be cal-
emax
culated
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
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 $< x >$
  2 ... Re $< y >$
  3 ... Re $< z >$
  4 ... Re $< x >$
  5 ... Re $< x >$

178
6. \text{Re} < y > < z > \\
7. \text{Im} < x > < y > \\
8. \text{Im} < x > < z > \\
9. \text{Im} < y > < z > \\

Options 7-9 apply only in presence of SO, options 4-6 only in non-orthogonal cases.

**line 5:** free format

<table>
<thead>
<tr>
<th>IMME</th>
<th>OFF/ON; optionally prints unsquared momentum matrix elements to unit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>NATOMS</td>
<td>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 $\epsilon_2$ 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 $\epsilon$. 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).</td>
</tr>
</tbody>
</table>

**line 6:** (optional) free format

| IATOMS | List of NATOMS atoms for which the opt. matrix elements should be calculated (see above). |

**8.18 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 equilibrium parameters (see Sec.5.3.1).

**8.18.1 Execution**

The program `optimize` is executed by invoking the command:

```
optimize optimize.def or x optimize
```

**8.18.2 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.
8.19 QTL (calculates special partial charges and population matrices)

This program was contributed by:

P. Novák and J. Kuneš
Inst. of Physics, Acad. Science, Prague, Czech 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 (with a limit of 28 different 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.qtltex [up/dn] is created in which the ordering of partial charges is given. Please note, that in contrast to case.qtl from x lapw2 -qt1 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 interstitial charge will usually NOT correct.

8.19.1 Execution

The program qtl is executed by invoking the command:

```
x qtl [ -up/dn -so -p -hf] or
qtl qtl.def
```

8.19.2 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 ------------------------
```
Table 8.93: Possible values of QSPLIT and their interpretation

<table>
<thead>
<tr>
<th>QSPLIT</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>DOS in basis according to ISPLIT from case.struct</td>
</tr>
<tr>
<td>-1</td>
<td>DOS in relativistic $</td>
</tr>
<tr>
<td>0</td>
<td>DOS in relativistic $</td>
</tr>
<tr>
<td>1</td>
<td>DOS in $</td>
</tr>
<tr>
<td>2</td>
<td>DOS in basis of real orbitals (no symmetry)</td>
</tr>
<tr>
<td>3</td>
<td>axial symmetry</td>
</tr>
<tr>
<td>4</td>
<td>hexagonal symmetry</td>
</tr>
<tr>
<td>5</td>
<td>cubic symmetry</td>
</tr>
<tr>
<td>6</td>
<td>user written unitary transformation</td>
</tr>
<tr>
<td>88</td>
<td>population matrix, no $&lt;l</td>
</tr>
<tr>
<td>99</td>
<td>full population matrix including $&lt;l</td>
</tr>
</tbody>
</table>

Interpretive comments on this file are as follows:

**line 1:** free format  
emin,emax                            energy window

**line 2:** free format  
natom                                number of atoms selected for calculation (max. 28,  
if more are needed you have to run qtl in “junks”)

**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

hz, 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 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 \( l_m \)-basis to the desired one. A few examples (e.g. \( j_j \), \( l_m \), or \( e_\gamma - \ell_\gamma \)) are supplied with the code in \$WIENROOT/SRC/templates/template.cf\$+ and must be copied to case.cf\$i. For less common cases these must be generated by hand.

8.19.3 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 interstitial 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:

```
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,px,py,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 interstitial 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.

8.20 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) or case.energy (with switch -enefile). 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 7.8). 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.

### Execution

The program spaghetti is executed by invoking the command:

```
spaghetti spaghetti.def or x spaghetti [-up|dn] [-so] [-p] [-hf] [-enefile]
```

The -p switch directs spaghetti to use the case.output1* files of a k-point parallel lapw1.

### 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, jcol, size of heavier plotting
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

**line 1:** free format
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
CHAPTER 8. ANALYSIS, PROPERTIES AND OPTIMIZATION

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 defs.f

**line 6: free format**
linew, ilin, icol

linew: line width
illin: 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

*nb*band1, *nb*band2 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.eña (which can be used for plotting with an external xy-plotting program).

**line 11:** free format

*ja*tom, *jc*ol, *jsize*

*ja*tom If a case.qtl file is present, *ja*tom indicates the atom whose character (selected by *jc*ol) is used for “band-character plotting” (dots are replaced by circles with radii proportional to the corresponding weight, requires *i*lin=0,2,3). If set to zero or if case.qtl is not present, “band-character plotting” does not occur.

*jc*ol 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 *ja*tom=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 *jc*ol and *jsize*).

---

### 8.21 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 $4\pi$, 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.

### 8.21.1 Execution

**Execution**

The program `telnes3` is executed by invoking the command:

```
  telnes3 telnes3.def or x telnes3 [-up|-dn]
```

### 8.21.2 Input

TELNES3 requires one input file - `case.innes`. We recommend using `InnesGen` 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:
<table>
<thead>
<tr>
<th>line</th>
<th>value</th>
<th>explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'Graphite ...'</td>
<td>Title (of no consequence for the calculation)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Atom number as given in case.struct (the index which numbers inequivalent atoms)</td>
</tr>
<tr>
<td>3</td>
<td>1 0</td>
<td>main and orbital quantum number n and l of the core state; eg. 1 0 stands for 1 s</td>
</tr>
<tr>
<td>4</td>
<td>285</td>
<td>energy of the edge onset in eV (here for the C K edge)</td>
</tr>
<tr>
<td>5</td>
<td>300</td>
<td>beam energy in keV</td>
</tr>
<tr>
<td>6</td>
<td>0.0 20.0 0.1</td>
<td>energy mesh given as $E_{min}E_{max}E_{step}$; all values in eV. 0.0 is the edge threshold.</td>
</tr>
<tr>
<td>7</td>
<td>5.0 1.87</td>
<td>detector collection semiangle and microscope convergence semiangle in mrad</td>
</tr>
<tr>
<td>8</td>
<td>10 1</td>
<td>parameters NR and NT which determine the mesh used for sampling the distribution of Q-vectors allowed by collection and convergence angles</td>
</tr>
<tr>
<td>9</td>
<td>0.8</td>
<td>spectrometer broadening FWHM in eV</td>
</tr>
<tr>
<td>10</td>
<td>END</td>
<td>keyword telling the program that there is no more input to read. Optional keywords and values must be inserted before this line!</td>
</tr>
</tbody>
</table>

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™ 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 containing 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 \( \alpha \) and convergence angle \( \beta \) 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 \(NR^2 \times NT\) points in total.
8.21. TELNES3

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 \( NR, \alpha \) and \( \beta \).

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.

Orientational Sensitive

\[ g_1 \ g_2 \ g_3 \]  
(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). The Euler angles \((0,0,0)\) means that the electron beam is parallel to the c-axis of the crystal and the 3 angles rotate with respect to the x-, y- and z-axes, respectively. Most likely, this option needs larger \( NR \) (and \( NT \)). If the ORIENTATION SENSITIVE key is not set, the program will average over all orientations (default).

Selection Rule

\[ \text{type} \]  
(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

\[ \text{type} \]  
(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 \( l \)):

- \( m \) : \( L=1 \)
- \( d \) : \( L=1 \) +/- 1
- \( q \) : \( L=1 \) +/- 2
- \( o \) : \( L=1 \) +/- 3
- \( n \) : no selection rule, calculate all transitions
- \( 0-3 \) : \( |L-l| \leq \text{type} \)
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**

R$_{\text{max}}$ sampling l$_{\text{max}}$ refine (e.g.: 3.0 15 0 1.0) (no defaults)

Calculate matrix elements beyond the muffin tin radius up to $r = \text{r}_{\text{max}}$ (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=l$_{\text{max}}$. 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.

### 8.21.3 Practical considerations

A typical ELNES calculation consists of the following steps:

- initialize \((\text{init\_lapw})\) and converge a SCF calculation \((\text{run\_lapw})\)
- provide a suitable \text{case\_innes} file
- if more excited states are needed than given by the SCF calculation, raise the upper energy limit in \text{case\_inl} and run \text{x\_lapw1}
- create the \text{case\_qtl} file using \text{x\_qtl -telnes}
- calculate the EELS spectrum using \text{x\_telnes3}. It is generally a smart move to make the program calculate the DOS on the biggest energy grid you will ever need, save this to file, and simply read it from file for all future calculations (INITIALIZATION key). This saves time. (In case of disk space problems, once the \text{case\_qtl} file has been created, the \text{case\_vector} files can be deleted. Similar, the \text{case\_qtl} file can be deleted or compressed once the \text{case\_dos} file exists.)
- add broadening to the spectrum using \text{x\_broadening}. If you wish, editing the \text{case\_inb} file allows tweaking of the broadening.
- study the output \((\text{case\_elnes} \text{ or case\_broadspec} \text{ are the place to start})\).
- if you wish to do more calculations, save the current results using \text{save\_eels -d calculation1}. Edit \text{case\_innes} and run \text{x\_telnes3} again.

This sequence can conveniently be executed using \text{w2web} by simply clicking one button after the other.

### 8.21.4 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 \text{case\_innes}. We list here all files possibly used by TELNES3 (and listed in \text{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.

- \text{case\_innes} (I). Defines the ELNES calculation. Always read.
- \text{case\_struct} (I). Defines the crystal. Always read.
- \text{case\_vsp} (I). Spherical component of the crystal potential. Read unless core and final state wavefunctions are read from file.
- \text{case\_vtotal} (I). Total crystal potential (can be generated by \text{lapw0}). Read if EXTEND POTENTIAL is used.
- \text{case\_rotij} (I). Rotation matrices that transform q-vectors between equivalent atoms. Read if INITIALIZATION tells the program to do so.
- \text{case\_dos} (IO). I-resolved density of states. Read or written depending on INITIALIZATION settings.
CHAPTER 8. ANALYSIS, PROPERTIES AND OPTIMIZATION

- case.xdos (IO). lm,l’m’-resolved density of states. Read or written depending on INITIAL-
  IZATION 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 calcu-
  lated.
- 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 sen-
  sitive.
- case.corewavef (O). Contains core wavefunctions. Written if core wavefunctions were calcu-
  lated and verbosity > 1.
- case.final (O). Contains APW radial basis functions for final states at selected energies. Writ-
  ten if verbosity > 1.
- case.ortho (O). Contains APW radial basis functions for final states at selected energies. Writ-
  ten 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 sensi-
  tive 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.error (O). Error file containing current error message; empty after successful calcula-
  tion. Always written.

8.22 TETRA (density of states)

This program calculates total and partial density of states (DOS) by means of the modified tetra-
hedron 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). The broadening schemes 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 the programs lapw2 (switch QTL) or qtl
and generates the DOS in states/Ry(cell (files case.dos1/2/3/... )) and in states/eV(cell (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).

Alternatively and for the total DOS only, you can use the switch -enefile which does not re-
quire case.qtl, but uses case.energy and case.scf2 (in case of parallel lapw1 use “cat
case.energy_1 case.energy_2 ... case.energy”).

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 8.19).

You can also direct *tetra* to sum-up some partial DOS components into a single DOS. This is for instance useful to sum over the different positions of one element.

Using the switches `-rxesw 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 5.10.5) or *Cgrace.dos.lapw* (see 5.10.6).

It is strongly recommended that you use “Run Programs [ ] Tasks [ ] Density of States” from *w2web*.

### 8.22.1 Execution

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

```
tetra tetra.def or x tetra [-up|dn] -enefile -so -hf -rxes -rxesw E1 E2
```

### 8.22.2 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)

### 8.22.3 Input

The required input file *case.int* can optionally be created using the *w2web* interface or the `configure_int.lapw` script (see 5.2.9).

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
SUM: 1 2 # NUMBER OF SUMMATIONS, max-nr-of summands
2 3 # this sums dos-cases 2+3 from the input above
------------------- bottom of file ------------------------
```

Interpretive comments on this file are as follows:

**line 1**: free format
- `title`
CHAPTER 8. ANALYSIS, PROPERTIES AND OPTIMIZATION

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.

twid 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

line 5: optional line (free format)
SUM, nsum, isummax

SUM the keyword SUM directs tetra to add-up some partial DOS specified in the lines above and produce case.dossum and case.dossumev.
nsum number of summations as specified below (max 7).
isummax max number of summands in the lines below.

line 6: optional line (free format)
iline1 iline2 ...

iline1,2,.. gives the DOS-cases which should be summed up (max isummax cases)
8.23  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., 1975, Schwarz et al., 1979, Schwarz and Wimmer, 1980].
For a brief introduction see below. It uses the partial charges in case.qtl. This file must be gen-
erated separately using lapw2. Partial densities of states in case.doslev 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.

8.23.1  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 spec-
tra are generated using the partial densities of states (in the case.doslev file) and multi-
plying 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]
If you want “orientation” sensitive XSPEC (like p-parallel and p-normal spectra, you may change in \texttt{case.int} the column-number to eg. \(p_x\) or \(p_z\) and rerun the last tree steps of the script above manually.

### 8.23.2 Dimensioning parameters

The following dimensioning parameters are collected in the files \texttt{param.inc} of \texttt{SRC_txspect} and \texttt{SRC_lorentz}:

- \texttt{IEMAX0} maximum number of energy steps in the spectrum (\texttt{SRC_lorentz})
- \texttt{NRAD} number of radial mesh points
- \texttt{LMAX} highest \(l+1\) in basis function inside sphere (consistent with input in \texttt{case.in1})

### 8.23.3 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  (1 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)
0.5  (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  (1 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

- \texttt{TITLE} Title

**line 2:** free format

- \texttt{NATO} Number of the selected atom (in \texttt{case.struct} file)
8.23. XSPEC

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:

<table>
<thead>
<tr>
<th>Spectrum</th>
<th>n</th>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LII,III</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 8.116: 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. LII and LIII 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, $E_0$, generated automat-
ically if AUTO was set in line 10
Absorption spectra: not used

line 13: free format

E1 Emission spectra: onset energy for broadening, $E_1$, generated automat-
ically if AUTO was set in line 10
Absorption spectra: not used

line 14: free format

E2 Emission spectra: onset energy for broadening, $E_2$, generated automat-
ically if AUTO was set in line 10
Absorption spectra: not used
9 Utility Programs

Contents

9.1 symmetso ................................................. 199
9.2 pairhess .............................................. 200
9.3 eigenhess .............................................. 202
9.4 patchsymm ............................................ 202
9.5 afminput .............................................. 203
9.6 clmcopy ................................................. 203
9.7 reformat ............................................... 205
9.8 hex2rhom and rhomb_in5 ......................... 205
9.9 plane .................................................. 205
9.10 add_columns ......................................... 206
9.11 add_columns_new .................................. 206
9.12 clminter ............................................. 206
9.13 eosfit ................................................. 206
9.14 calLa_Pre ............................................ 207
9.15 eosfit6 ............................................... 207
9.16 spacegroup ........................................... 207
9.17 join_vectorfiles .................................... 207
9.18 arrows ............................................... 208
9.19 xyz2struct .......................................... 209
9.20 cif2struct ........................................... 209
9.21 Tmaker ............................................... 210
9.22 struct2cif ........................................... 210
9.23 struct2poscar ....................................... 210
9.24 conv2prim ........................................... 210
9.25 fleur2wien ......................................... 210
9.26 StructGen of w2web ............................ 211
9.27 supercell ............................................. 211
9.28 structeditor ......................................... 211
9.29 Visualization ....................................... 213
9.30 Unsupported software ......................... 214

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 $\hat{A}$, 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- and B-type and throw away all C-type symmetry 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` 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 `.minhess`, 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:

```plaintext
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 strengths. 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:

\texttt{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 \texttt{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 (e.g. "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:

\texttt{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 non-magnetic 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 symmetry operations

9.6.3 Input

An example is given below:

```
---------------- top of file: case.inclmcopy ----------------
2
1 2  
-1.0000000000 0.0000000000 0.0000000000  
0.0000000000 -1.0000000000 0.0000000000  
0.0000000000 0.0000000000 -1.0000000000  
0
3 4  
-1.0000000000 0.0000000000 0.0000000000  
0.0000000000 -1.0000000000 0.0000000000  
0.0000000000 0.0000000000 -1.0000000000  
0
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  
9 1 0 0 0.50000  
0 0 1 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)
9.7. **REFORMAT**

**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  # length (in bohr) of plot in x and y direction.
'P'  # defines lattice, either P (cartesian 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  **add_columns_new**

`add_columns_new` adds or subtracts two columns of data from two files You have to specify the filenames and columns you want to add on the commandline like:

```
add_columns_new file1 col1 file2 col2 [add/sub]
```

The source of this program is supplied in `SRC_trig`.

9.12  **clminter**

`clminter` interpolates the density in `case.clmsum/up/dn` to a new radial mesh as defined in `case.struct_new`. This utility is useful 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 coming 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`.

9.13  **eosfit**

Small program to calculate the Equation of States (EOS; Equilibrium volume $V_0$, Bulk modulus $B_0$ and its derivative $B_0'$. The Murnaghan [Murnaghan, 1944], the Birch-Murnaghan, the EOS2, Vinet-Rose[Vinet et al., 1989] and Poirier-Tarantola[Poirier and Tarantola, 1998] 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 5.10.1 and 5.3.1).

The sources are supplied in `SRC.eosfit`.

### 9.14 calLa_Pre

Small program to calculate lattice parameters at a given pressure using `case_initial.struct` and `case.outpuseos` (both are created by a standard wien2k volume optimization) at constants b/a and c/a ratio

Contributed by Morteza Jamal (m.jamal57@yahoo.com).

### 9.15 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`.

### 9.16 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`.

### 9.17 join_vectorfiles

This program was contributed by:
Interactive program to combine parallel vector and energy files (\texttt{case.vector.xx} and \texttt{case.energy.xx}) into single files (\texttt{case.vector} and \texttt{case.energy}).

Executed by:

\begin{verbatim}
▶ x join_vectorfiles [-up/-dn/-so/-soup/-sodn] [-c] [-hf] case number_of_parallel_files_to_join
\end{verbatim}

## 9.18 arrows

This program was contributed by:

Evgeniya Kabliman
Institute for Materials Chemistry
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 \texttt{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 \texttt{forces} is:

\begin{itemize}
  \item Prepare (copy) a struct and scf file with the initial structure using the names \texttt{case_initial.struct} and \texttt{case_initial.scf}.
  \item View \texttt{case_initial.struct} in \texttt{Xcrysden} and “File/Save as xsf-structure” with the name \texttt{case_initial.xsf}.
  \item \texttt{x arrows}
  \item View the resulting \texttt{case_forces.xsf} using: \texttt{xcrysden --xsf case_forces.xsf}. Switch on “Display/Forces” and adjust the length of the arrows in “Modify/Force-settings”.
\end{itemize}

while differences between the initial and relaxed structure can be viewed by:

\begin{itemize}
  \item Prepare (copy) two struct files with the initial and the relaxed structure using the names: \texttt{case_initial.struct} and \texttt{case_final.struct}.
  \item View \texttt{case_initial.struct} in \texttt{Xcrysden} and “File/Save as xsf-structure” with the name \texttt{case_initial.xsf}.
  \item \texttt{x -delta arrows}
  \item View the resulting \texttt{case_delta.xsf} using: \texttt{xcrysden --xsf case_delta.xsf}. Switch on “Display/Forces” and adjust the length of the arrows in “Modify/Force-settings”.
\end{itemize}
9.19  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 cart. 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:

\texttt{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 \texttt{StructGen} and copy/paste of the positions).

A proper \texttt{case.xyz} file looks like:

\begin{verbatim}
ang
7.47700 7.47700 7.47700
B 4.98406667 1.24616667 0.0000000
C 6.23083333 2.49233333 0.0000000
.....
\end{verbatim}

or

\begin{verbatim}
BOHR F L
17.47700
0.470724637 0.492808141 0.0000000
-0.471118220 0.493012774 0.0000000
0.00000000 0.00000000 0.680559876
B 0.00000000 0.00000000 0.0000000
C 0.14300000 0.14300000 0.2500000
.....
\end{verbatim}

9.20  cif2struct

cif2struct reads structural data in cif-format from case.cif and writes them into case.struct. It is executed using:

\texttt{cif2struct case.cif or cif2struct case.txt or x cif2struct [-txt]}

The required cif files can be for example be obtained from Crystallographic databases (e.g. the Inorganic Crystal Structure DataBase ICSD) or from other programs (when transferred from MS-Windows, make sure to have it in “Unix-mode”, not in “Dos-mode”; eventually use dos2unix).

Alternatively, cif2struct can work with case.txt, which contains the following data:

\begin{verbatim}
a 0.0 0.0 0.0 # a..Ang, b..Bohr
4.755 4.755 12.991 90. 90. 120. # a,b,c,angles
'Al'
0.0000000 0.0000000 0.3520000 # atom-name
'O'
0.3063000 0.0000000 0.2500000 # atomic position
...
\end{verbatim}
9.21 Tmaker

Tmaker creates a struct-file init.struct from a file datastruct, which can be created by the script makestruct.lapw. It is executed using:

Tmaker

It was contributed by Morteza Jamal (m.jamal57@yahoo.com).

9.22 struct2cif

struct2cif creates a cif-file case.cif from case.struct. It is executed using:

x struct2cif or struct2cif struct2cif.def

It was contributed by F. Boucher (Florent.Boucher@cnrs-imn.fr) and L.D.Marks (L-marks@northwestern.edu). In order to work properly, the case.struct file should have a spacegroup label included. There is also a similar program struct2xyz available.

9.23 struct2poscar

struct2poscar creates the files case.poscar and case.xyz from case.struct. case.poscar and case.xyz are files which are used by the package dftd3 when periodic boundary conditions are switched on or off, respectively. It is executed using:

x struct2poscar or struct2poscar struct2poscar.def

9.24 conv2prim

conv2prim creates the file case_prim.struct which corresponds to the primitive cell of the conventional unit cell specified by case.struct.

It is executed using:

x conv2prim or conv2prim conv2prim.def

9.25 fleur2wien

fleur2wien converts the FLEUR-file which contains the exchange-correlation potential (case.potential) into the WIEN2k-format (case.r2v(dn)). The FLEUR-file case.lattice.harmonics, which contains the linear combinations of spherical harmonics, is also necessary. If the FLEUR and WIEN2k Bravais matrices are not the same, then the FLEUR direct Bravais matrix has to be specified at the beginning of case.lattice.harmonics below the keyword bravais (a, b, c lattice parameters specified at the 1st, 2nd, 3rd lines, respectively).

It is executed using:

x fleur2wien or fleur2wien fleur2wien.def
9.26 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 9.16 to generate equivalent positions)
- built in calculator to carry out simple arithmetic operations to specify the position parameters (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. \(-y, x, -z + 1/2\)).

9.27 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° 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 useful if you want to arrange the atoms in a certain way, e.g. 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.9.28).

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, ....

9.27.1 Execution

The program supercell is executed by invoking the command:

```
  supercell or x supercell
```

9.28 structeditor

This program was contributed by:
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 (matlab) routines and some fortran code, thus it requires octave (the free matlab version) and for visualization the xcrysden program.

A more extended documentation and some examples can be found in $WIENROOT/SRC/structeditor/doc, but the “most important” command helpstruct lists all available functions:

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
makeconventional * converting structure into the conventional form
makeprimitive * converting structure to the primitive form
makesupercell * creates supercell
makestructure * creates surface
mergestruct * merges two structures
movealla * moves all atoms with vector vec
replaceatom * replaces an atom with other atom
replaceeqatoms * replaces an atom and all equivalent with other atoms
rescale_c * rescales c for surface cell (vacume in the midle)
rescale_c_2 * rescales c for surface cell (vacume above)
rescale_c_3 * rescales c for surface cell (vacume audside)
rmatom * removes an atom
remeqatoms * removes an atom and all equivalent
rotateall * rotates all atoms around z with a given angle
rotateatomlist * rotates specified atoms around z with a given angle
rotatethreedim * rotates specified atom around vector with given angle
savestruct * saves crystal structure
shiftatomlist * shifts specified atoms by a vector
showequivalent * outputs list of equivalent atoms
showstructure * displays structure (using xcrysden)
smultatom * creates symmetry equivalent positions
sshift * symmetric shifts of equivalent atoms

You can get then specific help on a particular function using eg.:

help makesurface.
Note: Several routines (in particular makesupercell or makesurface) need a “conventional” cell as input. For all F, B or C-centered lattices you should first convert your structures using “makeconventional”.

PS: It is also fairly trivial to construct new functions starting from already existing ones or by combining them in a convenient way.

### 9.28.1 Execution

The **structeditor** is invoked within the octave environment and a typical sequence of commands could be:

```octave
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
# get help on the command makesupercell
help makesupercell
```

### 9.29 Visualization

#### 9.29.1 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/~hermann/Balsac/](http://www.fhi-berlin.mpg.de/~hermann/Balsac/)
9.29.2 XCrysDen

XCrysDen [Kokalj, 2003] 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 9.18)

![Figure 9.1: 3D electron density in TiC generated with XCrysDen](image)

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/

9.30 Unsupported software

On our website http://www.wien2k.at/reg_user 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)

\[ \text{TiC} \]
\[ \text{Fccni} \]
\[ \text{TiO2} \]

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] \(3d^{10}, 4s^2\). 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):
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 interstitial 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 ($\text{TiO}_2$)

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):

<table>
<thead>
<tr>
<th>Title</th>
<th>TiO2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spacegroup</td>
<td>$P4_2/mnm$ (136)</td>
</tr>
<tr>
<td>a</td>
<td>8.682 bohr</td>
</tr>
<tr>
<td>b</td>
<td>8.682 bohr</td>
</tr>
<tr>
<td>c</td>
<td>5.592 bohr</td>
</tr>
<tr>
<td>$\alpha, \beta, \gamma$</td>
<td>90</td>
</tr>
<tr>
<td>Atom</td>
<td>Ti, enter position (0,0,0) and RMT = 2.0</td>
</tr>
<tr>
<td>Atom</td>
<td>O, enter position (0.3,0.3,0) and RMT = 1.6</td>
</tr>
</tbody>
</table>

StructGen should automatically add the equivalent positions.

Initialize the calculation using RKmax=6.5 in tio2.in1 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 (\texttt{:ENE, :DIS, :FGL002}), either using “\texttt{Utils/Analysis}” in \texttt{w2web}, or ‘\texttt{grep :ENE tio2.scf}’. You should see some convergence of \texttt{: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 \texttt{w2web} using “\texttt{Execution} \texttt{[mini.positions]}”. This will generate \texttt{TiO2.inM}, and you can try option \texttt{PORT} with tolf=1.0 (instead of 2.0), otherwise stay with the default parameters. Repeat “\texttt{Execution} \texttt{[mini.positions]}” and start the minimization.

Alternatively you can use

\begin{verbatim}
min_lapw -p
\end{verbatim}

which is identical to:

\begin{verbatim}
min_lapw -j ‘‘run_lapw -I -fc 1 -p’’
\end{verbatim}

This will create \texttt{TiO2.inM} automatically, call the program \texttt{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 \texttt{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 \texttt{TiO2.inM} (change the method, reduce or increase the stepsize), and remove \texttt{TiO2.tmpM} (contains the “history” of the minimization and is used to calculate the velocities of the moving atoms).

(This should not be neccessary 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 \texttt{w2web}.

Create a new directory, copy the original TiC struct file into it and run \texttt{supercell} program:

\begin{verbatim}
mkdir super
cd super
cp ../TiC/TiC.struct .
x supercell
\end{verbatim}

Specify “\texttt{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).

\begin{verbatim}
cp TiC_super.struct super.struct
\end{verbatim}

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 changes. 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 or
http://www.wien2k.at/reg_user/textbooks/WIEN2k_lecture-notes_2013/
Part III

Installation of the WIEN2k package and Dimensioning of programs
11 Installation and Dimensioning

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, and Mac. Hardware requirements will change from case to case (small cases with 10 atoms per unit cell can be run on almost any PC under Linux), but we recommend a more powerful quad-core Intel-core2 PC with at least 4 GB (better 8-16GB GB) memory and plenty of disk space (a TB).

For coarse grain parallelization on the k-point level, a cluster of PCs with a Gb Ethernet 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
CHAPTER 11. INSTALLATION AND DIMENSIONING

- Tcl/Tk-Toolkit (for Xcrysden only)
- MPI+SCALAPACK (for fine grain parallelization only)
- ELPA (for fine grain parallelization only, optional)
- FFTW v.2 or 3 (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.

11.1.1 Installation tips for mpich, fftw (either version 2.1.5 or 3.3), and ELPA

This is only a brief guidance (assuming you use Intel’s mkl), you may need some Linux experience for this.

- Download the mpich1.2.7p1, fftw-2.1.5 / fftw-3.3, and ELPA sources from http://www.mpich.org/, http://www.fftw.org/download.html, and http://elpa.mpcdf.mpg.de (Please note, the fftw-3.x versions are incompatible with fftw-2.x, but both interfaces are available in WIEN2k using the -DFFTW2 or -DFFTW3 compiler option (in FOPT/FPOPT during siteconfig_lapw)
- unzip and untar the downloaded file
- Change into the expanded directories and configure the compilation.
  - Define your fortran compiler in the variables FC (elpa) or F77 (fftw) (setenv FC=ifort, or export FC=ifort) – and your c compiler analogously in case of ELPA installation.
  - Provide additional flags in case of ELPA:
    SCALAPACK_LDFLAGS must be set to “-L$MKLROOT/lib/intel64 -lmkl_core -lmkl_scalapack_lp64 -lmkl_intel_lp64 -lmkl_random -lmkl_blacs_intelmpi_lp64 -lpthread -WI,-rpath,$MKLROOT/lib/intel64”.
    FCFLAGS must be set to “-mkl”.
  - Use “--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.
  - Please refer to the ELPA documentation for additional information regarding openMP support!

- make
- make install (if you specified a “system-directory” like /usr/local you must have proper permissions for this step, eg. become root user)
- if not already done, add the binaries of your mpi-directory to your path (eg. for mpich: 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 for ifort12):

- cd $MKLROOT/interfaces/fftw2xf
- make libintel64 (or make libia32)
11.2 INSTALLATION OF WIEN2K

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 -DDFTW2 to FOPT in the Makefile of lapw0 and lapw2
add -lfftw2xf or -lfftw2xf-gnu to R_LIBS in the Makefile of lapw0 and lapw2

a similar procedure is available for fftw3 (just exchange above 2 by 3 in all statements)

This may speedup the fft-parts of these programs a bit.

11.1.2 Installation of LIBXC

In order to use the library of exchange and correlation functionals LIBXC [Marques et al., 2012], you need to install it and recompile lapw0. The steps are the following:

- Download LIBXC version 3.0.0 (or a more recent version if available, but not an older one) from http://octopus-code.org/wiki/Libxc:download
- Compile LIBXC with the same compiler that was used to compile WIEN2k. The basic steps are the following (example with ifort):
  - cd libxc-3.0.0/
  - ./configure FC=ifort --prefix=$LIBXCDIR
  - make
  - make check
  - make install

where LIBXCDIR is the directory where you choose to put the library and modules of LIBXC.

Then, lapw0 needs to be recompiled with additional options and flags in the Makefile (it can be set automatically in siteconfig):

- specify LIBXCDIR
- -I$(LIBXCDIR)/include/ -DLIBXC added to FOPT and FPOPT
- -L$(LIBXCDIR)/lib/ -lxcf03 -lxc added to LDFLAGS

Note that at the end of the compilation, the file $WIENROOT/SRC/lapw0/xc_funcs.h should be present.

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_12.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_amrit
  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_fsqen
  SRC_hf
  SRC_initxspec
  SRC_irrep
  SRC_joint
  SRC_kgen
  SRC_kram
  SRC_lapw0
  SRC_lapw1
  SRC_lapw2
  SRC_lapw3
  SRC_lapw4
  SRC_lapw7
  SRC_lapwmd
  SRC_lapwso
  SRC_lcore
  SRC_lib
  SRC_lorentz
  SRC_lstart
  SRC_mini
  SRC_mixer
  SRC_mn
  SRC_optic
  SRC_optimize
  SRC_orb
  SRC_pairhess
  SRC Phonon
  SRC_qt1
  SRC_reformat
  SRC_sgroup
  SRC_spacegroup
  SRC_spaghetti
  SRC_structeditor
  SRC_sumhfpara
  SRC_sumpara
  SRC_supercell
  SRC_symmetry
  SRC_symmetso
  SRC_tetelles3
  SRC_templates
  SRC_tetra
  SRC_trig
  SRC_txspec
  SRC_usersguide_html
  SRC_vegpratt
  SRC_w2web
  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.
```
11.2. INSTALLATION OF WIEN2k

/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 -llblas_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 whether you want to run fine-grained parallel. This is only possible if FFTW, MPI, SCALAPACK (included in newer mkl-versions), and, optionally, ELPA 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, lapw will start to “page” leading to unacceptable performance or a crash. NMATMAX will be automatically reduced (by $NPE/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)

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 corresponding versions with `make` and `make complex`, respectively. (The *.frc and *.F files will then be preprocessed automatically).

The fine-grained parallel versions `lapw0_mpi`, `lapwl_mpi`, `lapwl1c_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 `cputimGeneric.c` should work.

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 ‘‘name of compiler’’`) make these programs run faster. You could specify a higher optimization (-O3), or specify a particular processor architecture (`-arch=pwr5` or `-R10000, ...`).
- Good performance depends on highly optimized BLAS (and much less on LAPACK) libraries. Whenever possible, replace the supplied libraries (`SRC_lib/blas/lapw SRC_lib/lapack/lapw`), by routines from your vendor (mkl for Intel or AMD processors, acml for AMD, essl for IBM, sunperf for SUN, complib.sgmath on SGI, ...). Because of the superior performance of the Intel-mkl library we recommend ifort/mkl instead of gfortran (or some other commercial f90 compiler). If such libraries are not available use the OpenBLAS-library (`http://www.openblas.net/`). (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:

\[
\text{PARAMETER(XXXX = ....)}
\]

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

\[
\text{ssh -fNL 2000:w2web_host:7890 user@w2web_host}
\]

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

\[
\text{batch=batch < %f}
\]
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”

```bash
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:

```plaintext
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 set in your `.cshrc/.bashrc` file (by `userconfig_lapw`):

- **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. On slow NFS-filesystems, a “local” scratch-directory could greatly enhance the performance.
- **EDITOR**: path and name of your preferred 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 XCrys-Den.
- **BERRYPI_PATH**: path for the `BERRYPI` executables
- **BERRYPI_PYTHON**: path for `PYTHON` (at least 2.7)

The following variables are stored in `$WIENROOT/parallel_options` (created by `siteconfig_lapw`):

- **USE_REMOTE** [0/1] determines whether parallel jobs are run in background (on shared memory machines) or using ssh. It is usually set in `$WIENROOT/parallel_options`
- **MPI_REMOTE** [0/1] determines whether the mpirun command is issued on the “master-node”, or first an ssh to a remote node is done and there the mpirun command is issued. Usually, on many mpi-2 systems the first method is preferred, on mpi-1 the second.
- **WIEN_GRANULARITY**: Default granularity for parallel execution. It is overridden by setting the granularity in the `.machines`
- **WIEN_EXTRAFAINE**: if set, the residual k-points are spread one by one over the processors.
- **TASKSET** [no|command] specifies an optional command for binding a process to a specific core (like: `taskset -c`)
- **WIEN_MPIRUN**: is used to define the command to launch mpi-parallel calculations. This variable is set in `$WIENROOT/parallel_options` and usually is set to:
  
  "mpirun –np NP –machinefile HOSTS EXEC."

  In SLURM batch systems you may want to use:
  
  "srun –K –N nodes –n NP –r offset PINNING EXEC." and define in addition:
  
  **CORES_PER_NODE** (eg. 16 if you have 16 core nodes).
  
  **PINNING_COMMAND** (eg `–cpu_bind=map_cpu`).
  
  **PINNING_LIST** (eg: “0,8,1,9,2,10,3,11,4,12,5,13,6,14,7,15” on a 16 core machine). NOTE that for slurm only commensurate grids are supported at the moment (i.e., the number of processes of your grid is a multiple of - in case of small grids - a divisor of your CORES_PER_NODE).
In addition on some systems useful variables are:

**LD_LIBRARY_PATH** path to libraries of compiler and math-libs

**OMP_NUM_THREADS** [1—2—4] on multi-core machines for parallelization in certain libraries (mkl)
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 goto:
http://www.wien2k.at/reg_user/mailing_list/
and subscribe. You will then automatically be added to the mailing list
wien@theochem.tuwien.ac.at
and can post questions. 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 $w(r, E)$. 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 occuered. 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.ini. 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.ini and case.ini.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.ini (see sec. 7.5.3) 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 case.in1) 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 7.5.3).

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”.
12.1. 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 $R_{K\text{max}}$ for the cation than for H, (e.g. 16.0 when you select $R_{K\text{max}}=4.0$ in `case.in1`). This enormous difference in the convergence may lead to unphysical eigenvalues. In such cases choose $l_{\text{max}}=12$ 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.

---

```plaintext
----------------- 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

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 .3286919
.3477629 .3809219 .5143729 .5550735 .5550735
.5617155 .5617155 .7087550 .7197110 .8736991
.8736991 .9428865 .9533619 1.2224570 1.2224570
1.4285169

********************************************************
NUMBER OF K-POINTS: 1
```
---
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.

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:

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!
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 7.55 (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 \( R \), which transforms the global coordinates \( r \) to the rotated ones \( r' \), is defined by \( Rr = r' \).

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 8.13)
- 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:

\[
\begin{align*}
\text{Titanium} & \quad \text{operation} \# \quad 1 \quad 1 \\
\text{Titanium} & \quad \text{operation} \# \quad 2 \quad -1 \\
\text{Titanium} & \quad \text{operation} \# \quad 5 \quad 2 \parallel z \\
\text{Titanium} & \quad \text{operation} \# \quad 6 \quad m \ n \ z \\
\text{Titanium} & \quad \text{operation} \# \quad 12 \quad m \ n \ 110 \\
\text{Titanium} & \quad \text{operation} \# \quad 13 \quad m \ n \ -110 \\
\text{Titanium} & \quad \text{operation} \# \quad 18 \quad 2 \parallel 110 \\
\text{Titanium} & \quad \text{operation} \# \quad 19 \quad 2 \parallel -110 \\
\text{pointgroup} & \quad \text{is} \quad \text{mmm} \quad (\text{neg. iatnr!!}) \\
\text{axes} & \quad \text{should be:} \quad m \ n \ z, \ m \ n \ y, \ m \ n \ x
\end{align*}
\]

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 \(45^\circ\) around the \(z\) axis. (The required choice of the coordinate system for mmm symmetry is also given in Table 7.55)

\[
\begin{align*}
\text{Oxygen} & \quad \text{operation} \# \quad 1 \quad 1 \\
\text{Oxygen} & \quad \text{operation} \# \quad 6 \quad m \ n \ z \\
\text{Oxygen} & \quad \text{operation} \# \quad 13 \quad m \ n \ -110 \\
\text{Oxygen} & \quad \text{operation} \# \quad 18 \quad 2 \parallel 110 \\
\text{pointgroup} & \quad \text{is} \quad \text{mm2} \quad (\text{neg. iatnr!!}) \\
\text{axes} & \quad \text{should be:} \quad 2 \parallel z, \ m \ n \ y
\end{align*}
\]

For O the 2-fold symmetry axes points into the \((110)\) direction instead of \(z\). The appropriate rotation matrices for Ti and O are:

\[
\begin{pmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\
0 & 0 & 1
\end{pmatrix} \quad \begin{pmatrix}
0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\
0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{pmatrix}
\]

A.2 Si \(\Gamma\)-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 \(\Gamma\)-phonon 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:

\[
\begin{align*}
\text{Si} & \quad \text{operation} \# \quad 1 \quad 1 \\
\text{Si} & \quad \text{operation} \# \quad 13 \quad m \ n \ -110 \\
\text{Si} & \quad \text{operation} \# \quad 16 \quad m \ n \ -101 \\
\text{Si} & \quad \text{operation} \# \quad 17 \quad m \ n \ 0-11 \\
\text{Si} & \quad \text{operation} \# \quad 24 \quad 3 \parallel 111 \\
\text{Si} & \quad \text{operation} \# \quad 38 \quad 3 \parallel 111 \\
\text{pointgroup} & \quad \text{is} \quad \text{3m} \quad (\text{neg. iatnr!!}) \\
\text{axis} & \quad \text{should be:} \quad 3 \parallel z, \ m \ n \ y
\end{align*}
\]

\(\text{lm:} \quad 0 \ 0 \ 1 \ 0 \ 2 \ 0 \ 3 \ 0 \ 3 \ 3 \ 4 \ 0 \ 4 \ 3 \ 5 \ 0 \ 5 \ 3 \ 6 \ 0 \ 6 \ 3 \ 6\)
Therefore the required local rotation matrix should rotate \( z \) into the (111) direction and thus the matrix in the “struct” file should be:

\[
\begin{bmatrix}
0.4082483 & -0.7071068 & 0.5773503 \\
0.4082483 & 0.7071068 & 0.5773503 \\
-0.8164966 & 0.0000000 & 0.5773503
\end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\
\frac{\sqrt{6}}{6} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\
-\frac{2}{\sqrt{6}} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3}
\end{bmatrix}
\]

### 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
0.0000000 0.0000000 0.0
6 IORD OF GROUP G0
```

The output of SYMMETRY reads:

```
Se operation # 1 1
Se operation # 9 2 $|$ $|$ $|$ $|$ 110
point group is 2 (neg. iatnr!!)
axis should be: 2 || z
lm: 0 0 1 0 2 0 2 2 2 2 2 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

<table>
<thead>
<tr>
<th>H</th>
<th>Li</th>
<th>Be</th>
<th>B</th>
<th>C</th>
<th>N</th>
<th>O</th>
<th>F</th>
<th>Ne</th>
<th>/*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Na</td>
<td>Mg</td>
<td>Al</td>
<td>Si</td>
<td>P</td>
<td>S</td>
<td>Cl</td>
<td>Ar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Ca</td>
<td>Sc</td>
<td>Ti</td>
<td>V</td>
<td>Cr</td>
<td>Mn</td>
<td>Fe</td>
<td>Co</td>
<td>Ni</td>
</tr>
<tr>
<td>Rb</td>
<td>Sr</td>
<td>Y</td>
<td>Zr</td>
<td>Nb</td>
<td>Mo</td>
<td>Tc</td>
<td>Ru</td>
<td>Rh</td>
<td>Pd</td>
</tr>
<tr>
<td>Cs</td>
<td>Ba</td>
<td>Hf</td>
<td>Ta</td>
<td>W</td>
<td>Re</td>
<td>Os</td>
<td>Ir</td>
<td>Pt</td>
<td>Au</td>
</tr>
<tr>
<td>Fr</td>
<td>Ra</td>
<td>Ac</td>
<td>Lr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>He</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>H</th>
<th>Li</th>
<th>Be</th>
<th>B</th>
<th>C</th>
<th>N</th>
<th>O</th>
<th>F</th>
<th>Ne</th>
<th>/*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Na</td>
<td>Mg</td>
<td>Al</td>
<td>Si</td>
<td>P</td>
<td>S</td>
<td>Cl</td>
<td>Ar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Ca</td>
<td>Sc</td>
<td>Ti</td>
<td>V</td>
<td>Cr</td>
<td>Mn</td>
<td>Fe</td>
<td>Co</td>
<td>Ni</td>
</tr>
<tr>
<td>Rb</td>
<td>Sr</td>
<td>Y</td>
<td>Zr</td>
<td>Nb</td>
<td>Mo</td>
<td>Tc</td>
<td>Ru</td>
<td>Rh</td>
<td>Pd</td>
</tr>
<tr>
<td>Cs</td>
<td>Ba</td>
<td>Hf</td>
<td>Ta</td>
<td>W</td>
<td>Re</td>
<td>Os</td>
<td>Ir</td>
<td>Pt</td>
<td>Au</td>
</tr>
<tr>
<td>Fr</td>
<td>Ra</td>
<td>Ac</td>
<td>Lr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| He |

<table>
<thead>
<tr>
<th>H</th>
<th>Li</th>
<th>Be</th>
<th>B</th>
<th>C</th>
<th>N</th>
<th>O</th>
<th>F</th>
<th>Ne</th>
<th>/*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Na</td>
<td>Mg</td>
<td>Al</td>
<td>Si</td>
<td>P</td>
<td>S</td>
<td>Cl</td>
<td>Ar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Ca</td>
<td>Sc</td>
<td>Ti</td>
<td>V</td>
<td>Cr</td>
<td>Mn</td>
<td>Fe</td>
<td>Co</td>
<td>Ni</td>
</tr>
<tr>
<td>Rb</td>
<td>Sr</td>
<td>Y</td>
<td>Zr</td>
<td>Nb</td>
<td>Mo</td>
<td>Tc</td>
<td>Ru</td>
<td>Rh</td>
<td>Pd</td>
</tr>
<tr>
<td>Cs</td>
<td>Ba</td>
<td>Hf</td>
<td>Ta</td>
<td>W</td>
<td>Re</td>
<td>Os</td>
<td>Ir</td>
<td>Pt</td>
<td>Au</td>
</tr>
<tr>
<td>Fr</td>
<td>Ra</td>
<td>Ac</td>
<td>Lr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| He |

<table>
<thead>
<tr>
<th>H</th>
<th>Li</th>
<th>Be</th>
<th>B</th>
<th>C</th>
<th>N</th>
<th>O</th>
<th>F</th>
<th>Ne</th>
<th>/*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Na</td>
<td>Mg</td>
<td>Al</td>
<td>Si</td>
<td>P</td>
<td>S</td>
<td>Cl</td>
<td>Ar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Ca</td>
<td>Sc</td>
<td>Ti</td>
<td>V</td>
<td>Cr</td>
<td>Mn</td>
<td>Fe</td>
<td>Co</td>
<td>Ni</td>
</tr>
<tr>
<td>Rb</td>
<td>Sr</td>
<td>Y</td>
<td>Zr</td>
<td>Nb</td>
<td>Mo</td>
<td>Tc</td>
<td>Ru</td>
<td>Rh</td>
<td>Pd</td>
</tr>
<tr>
<td>Cs</td>
<td>Ba</td>
<td>Hf</td>
<td>Ta</td>
<td>W</td>
<td>Re</td>
<td>Os</td>
<td>Ir</td>
<td>Pt</td>
<td>Au</td>
</tr>
<tr>
<td>Fr</td>
<td>Ra</td>
<td>Ac</td>
<td>Lr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| He |

<table>
<thead>
<tr>
<th>H</th>
<th>Li</th>
<th>Be</th>
<th>B</th>
<th>C</th>
<th>N</th>
<th>O</th>
<th>F</th>
<th>Ne</th>
<th>/*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Na</td>
<td>Mg</td>
<td>Al</td>
<td>Si</td>
<td>P</td>
<td>S</td>
<td>Cl</td>
<td>Ar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Ca</td>
<td>Sc</td>
<td>Ti</td>
<td>V</td>
<td>Cr</td>
<td>Mn</td>
<td>Fe</td>
<td>Co</td>
<td>Ni</td>
</tr>
<tr>
<td>Rb</td>
<td>Sr</td>
<td>Y</td>
<td>Zr</td>
<td>Nb</td>
<td>Mo</td>
<td>Tc</td>
<td>Ru</td>
<td>Rh</td>
<td>Pd</td>
</tr>
<tr>
<td>Cs</td>
<td>Ba</td>
<td>Hf</td>
<td>Ta</td>
<td>W</td>
<td>Re</td>
<td>Os</td>
<td>Ir</td>
<td>Pt</td>
<td>Au</td>
</tr>
<tr>
<td>Fr</td>
<td>Ra</td>
<td>Ac</td>
<td>Lr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| He |
Bibliography


247
BIBLIOGRAPHY


[Desclaux, 1969] Desclaux, J. P. (1969). Comput. 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. 8, 105, 139


