

# WIEN2k

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

User's Guide, WIEN2k\_23.2 (Release 07/24/2024)

**Peter Blaha**  
**Karlheinz Schwarz**  
**Georg K. H. Madsen**  
**Dieter Kvasnicka**  
**Joachim Luitz**  
**Robert Laskowski**  
**Fabien Tran**  
**Laurence D. Marks**

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,  
Robert Laskowski, Fabien Tran, Laurence D. Marks:**

**WIEN2k**

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

*revised edition WIEN2k.23.2 (Release 07/24/2024)*

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

---

<b>I</b>	<b>Introduction to the WIEN2k package</b>	<b>1</b>
<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Basic concepts</b>	<b>7</b>
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 . . . . .	10
2.2.3	General considerations . . . . .	10
<b>3</b>	<b>Quick Start</b>	<b>13</b>
3.1	Naming conventions . . . . .	13
3.2	Starting the server . . . . .	14
3.3	Connecting to the <b>w2web</b> server . . . . .	15
3.4	Creating a new session . . . . .	15
3.5	Creating a new case . . . . .	15
3.6	Creating the struct file . . . . .	15
3.7	Initialization . . . . .	18
3.8	The SCF calculation . . . . .	21
3.9	The case.scf file . . . . .	23
3.10	Saving a calculation . . . . .	23
3.11	Calculating properties . . . . .	24
3.11.1	Electron density plots . . . . .	24
3.11.2	Density of States (DOS) . . . . .	28
3.11.3	X-ray spectra . . . . .	29
3.11.4	Bandstructure . . . . .	29
3.11.5	Bandstructure with band character plotting / full lines . . . . .	31
3.11.6	Volume Optimization . . . . .	31
3.12	Setting up a new case . . . . .	32
3.12.1	Manual setup . . . . .	32
3.12.2	Setting up a new case using <b>w2web</b> . . . . .	33

<b>II</b>	<b>Detailed description of the files and programs of the WIEN2k package</b>	<b>35</b>
<b>4</b>	<b>Files and Program Flow</b>	<b>37</b>
4.1	Flow of input and output files . . . . .	37
4.2	Input/Output files . . . . .	41
4.3	The case.struct.file . . . . .	42
4.4	The case.scf file . . . . .	46
4.5	Flow of programs . . . . .	48
4.5.1	Core, semi-core and valence states . . . . .	48
4.5.2	Spin-polarized calculation . . . . .	50
4.5.3	Fixed-spin-moment (FSM) calculations . . . . .	50
4.5.4	Staggered field inside atomic spheres to vary the magnetic moment . . . . .	50
4.5.5	Antiferromagnetic (AFM) calculations . . . . .	51
4.5.6	Spin-orbit interaction . . . . .	52
4.5.7	Orbital potentials . . . . .	52
4.5.8	Onsite-exact-exchange and hybrid functionals for correlated electrons . . . . .	53
4.5.9	Unscreened and screened hybrid functionals ("hf"-module) . . . . .	55
4.5.10	Slater, SmBJ and KLI potentials ("hf"-module) . . . . .	61
4.5.11	Modified Becke-Johnson potential (mBJ) for band gaps . . . . .	62
4.5.12	Local modified Becke-Johnson potential (lmBJ) for interfaces and systems with vacuum . . . . .	63
4.5.13	GLLB-SC method . . . . .	64
4.5.14	DFT-1/2 method . . . . .	64
4.5.15	DFT-D3 and DFT-D4 for dispersion energy . . . . .	66
4.5.16	Nonlocal van der Waals functionals . . . . .	66
4.5.17	Self-consistent gKS MGGA . . . . .	68
<b>5</b>	<b>Shell scripts</b>	<b>71</b>
5.1	Job control . . . . .	71
5.1.1	Main execution script (x_lapw) . . . . .	71
5.1.2	Create the master input file case.struct (makestruct.lapw) . . . . .	73
5.1.3	Job control for initialization (init_lapw) . . . . .	73
5.1.4	Job control for scf-iterations (run_lapw or runsp_lapw) . . . . .	75
5.1.5	Job for initialization and scf-cycle with different precision (run123.lapw) . . . . .	78
5.1.6	Job control for iteration with the Slater/SmBJ/KLI potentials (run_vnonloc.lapw) . . . . .	78
5.1.7	Job control for calculating the exchange discontinuity of the GLLB-SC method (run_deltagllb.lapw) . . . . .	79
5.2	Utility scripts . . . . .	80
5.2.1	Save a calculation (save.lapw) . . . . .	80

5.2.2	Restoring a calculation (restore_lapw)	80
5.2.3	Remove unnecessary files (clean_lapw)	81
5.2.4	Migrate a case to/from a remote computer (migrate_lapw)	81
5.2.5	Set R-MT values in your case.struct file (setrmt_lapw)	81
5.2.6	Generate case.inst (instgen_lapw)	82
5.2.7	Check for running WIEN jobs (check_lapw)	82
5.2.8	Cancel (kill) running WIEN jobs (cancel_lapw)	82
5.2.9	grepline_lapw	82
5.2.10	scfmonitor_lapw	83
5.2.11	analyse_lapw	83
5.2.12	Extract critical points from a Bader analysis (extractaim_lapw)	84
5.2.13	Check parallel execution (testpara_lapw)	84
5.2.14	Check parallel execution of lapw1 (testpara1_lapw)	84
5.2.15	Check parallel execution of lapw2 (testpara2_lapw)	84
5.2.16	Create case.int file (for DOS) (configure_int_lapw)	84
5.2.17	init_orb_lapw	85
5.2.18	init_so_lapw	85
5.2.19	init_hf_lapw	85
5.2.20	init_mbj_lapw	86
5.2.21	init_mgga_lapw	86
5.2.22	vec2old_lapw	86
5.2.23	joinvec_lapw	87
5.2.24	Reduce atomic spheres and interpolate density (reduce_rmt_lapw)	87
5.2.25	clmextrapol_lapw	87
5.2.26	create_add_atom_clmsum_lapw	88
5.3	Structure optimization	89
5.3.1	Lattice parameters (Volume, $c/a$ , lattice parameters)	89
5.3.2	Minimization of internal parameters	93
5.4	Phonon calculations	96
5.4.1	PHONON	96
5.4.2	PHONOPY	97
5.5	Parallel Execution	98
5.5.1	k-Point Parallelization	98
5.5.2	MPI parallelization	99
5.5.3	How to use <b>WIEN2k</b> as a parallel program	99
5.5.4	The <b>.machines</b> file	100
5.5.5	How the list of k-points is split	102
5.5.6	Flow chart of the parallel scripts	102

5.5.7	On the fine grained parallelization . . . . .	104
5.6	NMR calculations: Chemical shift and Knight shift . . . . .	106
5.6.1	Chemical shift . . . . .	106
5.6.2	Knight shifts . . . . .	110
5.7	Wannier functions (wien2wannier) . . . . .	112
5.7.1	Usage . . . . .	112
5.7.2	Help and FAQ . . . . .	114
5.8	Spontaneous Polarization, Piezoelectricity and Born Charges, Weyl points (BerryPI) . . . . .	114
5.8.1	Options . . . . .	114
5.8.2	Spontaneous Polarization . . . . .	115
5.8.3	Born effective charges . . . . .	115
5.8.4	Piezoelectric constants . . . . .	116
5.8.5	Weyl points . . . . .	117
5.9	Getting on-line help . . . . .	117
5.10	Interface scripts . . . . .	118
5.10.1	eplot_lapw . . . . .	118
5.10.2	gibbs_lapw . . . . .	118
5.10.3	parabolfit_lapw . . . . .	119
5.10.4	dosplot_lapw . . . . .	119
5.10.5	dosplot2_lapw . . . . .	119
5.10.6	Cgrace_lapw, Cgrace_conf_lapw and Cgrace_dos_lapw . . . . .	120
5.10.7	Curve_lapw . . . . .	120
5.10.8	specplot_lapw . . . . .	120
5.10.9	rhoplot_lapw . . . . .	120
5.10.10	prepare_xsf_lapw . . . . .	120
5.10.11	opticplot_lapw . . . . .	122
5.10.12	addjoint-updn_lapw . . . . .	122
5.10.13	create_elf_lapw . . . . .	122
<b>6</b>	<b>Initialization</b> . . . . .	<b>125</b>
6.1	NN . . . . .	125
6.1.1	Execution . . . . .	126
6.2	SGROUP . . . . .	126
6.2.1	Execution . . . . .	126
6.3	SYMMETRY . . . . .	126
6.3.1	Execution . . . . .	127
6.4	LSTART . . . . .	127
6.4.1	Execution . . . . .	127
6.4.2	Dimensioning parameters . . . . .	128

6.4.3	Input	128
6.5	KGEN	130
6.5.1	Execution	130
6.5.2	Dimensioning parameters	130
6.6	DSTART	130
6.6.1	Execution	131
6.6.2	Dimensioning parameters	131
<b>7</b>	<b>SCF cycle</b>	<b>133</b>
7.1	LAPW0	133
7.1.1	Execution	134
7.1.2	Dimensioning parameters	134
7.1.3	Input	135
7.2	DFTD3	144
7.2.1	Execution	144
7.2.2	Input	144
7.3	DFTD4	145
7.3.1	Execution	145
7.3.2	Input	145
7.4	NLVDW	146
7.4.1	Execution	146
7.4.2	Input	146
7.5	ORB	147
7.5.1	Execution	148
7.5.2	Dimensioning parameters	148
7.5.3	Input	148
7.6	LAPW1	151
7.6.1	Execution	151
7.6.2	Dimensioning parameters	152
7.6.3	Input	152
7.7	HF	156
7.7.1	Execution	156
7.7.2	Input	156
7.8	LAPWSO	158
7.8.1	Execution	158
7.8.2	Dimensioning parameters	159
7.8.3	Input	159
7.9	LAPW2	160
7.9.1	Execution	160

7.9.2	Dimensioning parameters . . . . .	161
7.9.3	Input . . . . .	161
7.10	SUMPARA . . . . .	165
7.10.1	Execution . . . . .	165
7.10.2	Dimensioning parameters . . . . .	165
7.11	LAPWDM . . . . .	166
7.11.1	Execution . . . . .	166
7.11.2	Dimensioning parameters . . . . .	167
7.11.3	Input . . . . .	167
7.12	LCORE . . . . .	167
7.12.1	Execution . . . . .	168
7.12.2	Dimensioning parameters . . . . .	168
7.12.3	Input . . . . .	168
7.13	MIXER . . . . .	169
7.13.1	Execution . . . . .	170
7.13.2	Dimensioning parameters . . . . .	170
7.13.3	Input . . . . .	170
<b>8</b>	<b>Analysis, Properties and Optimization</b>	<b>173</b>
8.1	afmsim . . . . .	174
8.1.1	Execution . . . . .	174
8.2	AIM . . . . .	174
8.2.1	Execution . . . . .	175
8.2.2	Dimensioning parameters . . . . .	175
8.2.3	Input . . . . .	175
8.3	BerryPI . . . . .	178
8.4	BROADENING . . . . .	178
8.4.1	Execution . . . . .	179
8.4.2	Input . . . . .	179
8.5	DIPAN . . . . .	180
8.5.1	Execution . . . . .	181
8.5.2	Dimensioning parameters . . . . .	181
8.5.3	Input . . . . .	181
8.6	ELAST . . . . .	182
8.6.1	Execution . . . . .	182
8.6.2	Input . . . . .	183
8.7	FILTVEC . . . . .	185
8.7.1	Execution . . . . .	186
8.7.2	Dimensioning parameters . . . . .	186



8.7.3	Input file case.inf . . . . .	186
8.8	FSGEN . . . . .	188
8.9	IRelast . . . . .	188
8.10	IRREP . . . . .	190
8.10.1	Execution . . . . .	190
8.10.2	Dimensioning parameters . . . . .	191
8.11	JOINT . . . . .	191
8.11.1	Execution . . . . .	191
8.11.2	Dimensioning parameters . . . . .	192
8.11.3	Input . . . . .	192
8.12	KRAM . . . . .	193
8.12.1	Execution . . . . .	194
8.12.2	Dimensioning parameters . . . . .	194
8.12.3	Input . . . . .	194
8.13	LAPW3 . . . . .	195
8.13.1	Execution . . . . .	195
8.13.2	Dimensioning parameters . . . . .	195
8.14	LAPW5 . . . . .	195
8.14.1	Execution . . . . .	196
8.14.2	Dimensioning parameters . . . . .	196
8.14.3	Input . . . . .	196
8.15	3DDENS . . . . .	199
8.15.1	Execution . . . . .	199
8.15.2	Dimensioning parameters . . . . .	199
8.15.3	Input . . . . .	199
8.16	LAPW7 . . . . .	201
8.16.1	Execution . . . . .	201
8.16.2	Dimensioning parameters . . . . .	202
8.16.3	Input . . . . .	202
8.17	MINI . . . . .	204
8.17.1	Execution . . . . .	204
8.17.2	Dimensioning parameters . . . . .	204
8.17.3	Input . . . . .	205
8.18	MSTAR . . . . .	206
8.18.1	Execution . . . . .	207
8.19	NMR . . . . .	207
8.20	OPTIC . . . . .	208
8.20.1	Execution . . . . .	209

8.20.2	Dimensioning parameters . . . . .	210
8.20.3	Input . . . . .	210
8.21	OPTIMIZE . . . . .	211
8.21.1	Execution . . . . .	211
8.21.2	Input . . . . .	212
8.22	PES . . . . .	212
8.22.1	Execution . . . . .	212
8.23	QTL . . . . .	213
8.23.1	Execution . . . . .	214
8.23.2	Input . . . . .	214
8.23.3	Output . . . . .	215
8.24	RENDOS . . . . .	216
8.24.1	Execution . . . . .	216
8.25	SPAGHETTI . . . . .	216
8.25.1	Execution . . . . .	217
8.25.2	Input . . . . .	217
8.26	TELNES3 . . . . .	219
8.26.1	Execution . . . . .	220
8.26.2	Input . . . . .	220
8.26.3	Practical considerations . . . . .	225
8.26.4	Files . . . . .	225
8.27	TETRA . . . . .	226
8.27.1	Execution . . . . .	227
8.27.2	Dimensioning parameters . . . . .	227
8.27.3	Input . . . . .	227
8.28	XSPEC . . . . .	229
8.28.1	Execution . . . . .	229
8.28.2	Dimensioning parameters . . . . .	230
8.28.3	Input . . . . .	230
<b>9</b>	<b>Utility Programs</b>	<b>233</b>
9.1	add_columns . . . . .	234
9.2	add_columns_new . . . . .	234
9.3	afminput . . . . .	234
9.3.1	Execution . . . . .	235
9.3.2	Dimensioning parameters . . . . .	235
9.4	animxf . . . . .	235
9.5	arrows . . . . .	235
9.6	calla_Pre . . . . .	236

9.7	cif2struct . . . . .	236
9.8	clminter . . . . .	236
9.9	clmcopy . . . . .	237
	9.9.1 Execution . . . . .	237
	9.9.2 Dimensioning parameters . . . . .	237
	9.9.3 Input . . . . .	237
9.10	conv2prim . . . . .	238
9.11	create_rho . . . . .	238
9.12	eigenhess . . . . .	239
9.13	eosfit . . . . .	239
9.14	eosfit6 . . . . .	239
9.15	fleur2wien . . . . .	239
9.16	hex2rhomb and rhomb_in5 . . . . .	240
9.17	join_vectorfiles . . . . .	240
9.18	pairhess . . . . .	240
	9.18.1 Execution . . . . .	241
	9.18.2 Dimensioning parameters . . . . .	241
	9.18.3 Input . . . . .	241
9.19	patchsymm . . . . .	242
	9.19.1 Execution . . . . .	242
9.20	plane . . . . .	242
9.21	read_vorb_files . . . . .	243
9.22	reformat . . . . .	243
9.23	spacegroup . . . . .	243
9.24	struct2cif . . . . .	244
9.25	struct2poscar . . . . .	244
9.26	structeditor . . . . .	244
	9.26.1 Execution . . . . .	245
9.27	<b>StructGen of w2web</b> . . . . .	246
9.28	supercell . . . . .	246
	9.28.1 Execution . . . . .	246
9.29	symmetso . . . . .	247
	9.29.1 Execution . . . . .	247
9.30	Tmaker . . . . .	247
9.31	Visualization . . . . .	247
	9.31.1 XCrysDen . . . . .	247
	9.31.2 VESTA . . . . .	248
	9.31.3 BALSAC . . . . .	248
9.32	xyz2struct . . . . .	249
9.33	Unsupported software . . . . .	250

<b>10 Examples</b>	<b>251</b>
10.1 TiC . . . . .	251
10.2 FCC Nickel . . . . .	251
10.3 Rutile . . . . .	252
10.3.1 using w2web and the recommended MSR1a minimization . . . . .	252
10.3.2 PORT minimization . . . . .	253
10.3.3 Command line usage: . . . . .	253
10.4 supercell calc . . . . .	254
10.5 Further examples . . . . .	255
<b>III Installation of the WIEN2k package and Dimensioning of programs</b>	<b>257</b>
<b>11 Installation and Dimensioning</b>	<b>259</b>
11.1 Requirements . . . . .	259
11.1.1 Installation tips for fftw3, ELPA and LIBXC . . . . .	260
11.2 Installation of <b>WIEN2k</b> . . . . .	262
11.2.1 Check the software requirements . . . . .	262
11.2.2 Expanding the <b>WIEN2k</b> distribution . . . . .	262
11.2.3 Site configuration for <b>WIEN2k</b> . . . . .	264
11.2.4 User configuration . . . . .	265
11.2.5 Performance and special considerations . . . . .	265
11.2.6 Global dimensioning parameters . . . . .	266
11.3 <b>w2web</b> . . . . .	266
11.3.1 General issues . . . . .	266
11.3.2 How does <b>w2web</b> work? . . . . .	267
11.3.3 <b>w2web</b> -files in you home directory . . . . .	267
11.3.4 The configuration file conf/w2web.conf . . . . .	267
11.3.5 The password file conf/w2web.users . . . . .	268
11.3.6 Using the https-protocol with <b>w2web</b> . . . . .	268
11.4 Environment Variables . . . . .	268
<b>12 Trouble shooting</b>	<b>271</b>
12.1 Ghost bands . . . . .	272
<b>IV Appendix</b>	<b>277</b>
<b>A Local rotation matrices</b>	<b>279</b>
A.1 Rutile ( $TiO_2$ ) . . . . .	280
A.2 Si $\Gamma$ -phonon . . . . .	280
A.3 Trigonal Selenium . . . . .	281





---

# List of Tables

---

4.1	Input and output files of init programs . . . . .	39
4.2	Input and output files of utility programs . . . . .	40
4.3	Input and output files of main programs in an SCF cycle . . . . .	41
4.4	Lattice type, description and bravais matrix used in <b>WIEN2k</b> . The angle $\gamma'$ is defined via $\cos(\gamma) = \cos(\gamma') \sin(\alpha) \sin(\beta) + \cos(\beta) \cos(\alpha)$ . . . . .	43
4.23	MGGA functionals with support for automatic setup. For an exhaustive and up-to-date list of all keywords, see the LibXC website <a href="https://libxc.gitlab.io/functionals/">https://libxc.gitlab.io/functionals/</a> . . . . .	69
6.6	Relativistic quantum numbers . . . . .	129
7.3	global XC-switches . . . . .	138
7.4	EX-switches . . . . .	138
7.5	EC-switches . . . . .	139
7.6	VX-switches . . . . .	141
7.7	VC-switches . . . . .	142
7.56	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. . . . .	164
7.57	LM combination and local coordinate system of “non-cubic groups” (requires “negative atomic index” in case.struct) . . . . .	164
8.101	Possible values of QSPLIT and their interpretation . . . . .	215
8.124	Quantum numbers of the core state involved in the x-ray spectra . . . . .	231





---

# List of Figures

---

2.1	Partitioning of the unit cell into atomic spheres (I) and an interstitial region (II) . . .	9
3.1	TiC in the sodium chloride structure. This plot was generated using XCrysDen (see 9.31.1), which shows up in <b>w2web</b> when xcrysden is properly installed. Alternatively, also VESTA (see 9.31.2) can plot WIEN2k-structures. . . . .	14
3.2	Startup screen of <b>w2web</b> . . . . .	16
3.3	Main window of <b>w2web</b> . . . . .	17
3.4	<b>StructGen</b> of <b>w2web</b> . . . . .	19
3.5	Initialization with <b>w2web</b> . . . . .	20
3.6	List of input files . . . . .	22
3.7	Task “Electron Density Plots” . . . . .	24
3.8	Electron density of TiC in (100) plane using Xcrysden . . . . .	26
3.9	Electron density of TiC in (100) plane . . . . .	27
3.10	Density of states of TiC . . . . .	28
3.11	Density of states of TiC . . . . .	29
3.12	Ti $L_{III}$ spectrum of TiC . . . . .	30
3.13	Bandstructure of TiC . . . . .	30
3.14	Bandstructure of TiC, showing t2g-character bands of Ti in character plotting mode .	31
3.15	Energy vs. volume curve for TiC . . . . .	32
4.1	Data flow during a SCF cycle (programX.def, case.struct, case.inX, case.outputX and optional files are omitted) . . . . .	38
4.2	Program flow in <b>WIEN2k</b> . . . . .	49
5.1	Flow chart of <b>lapw1para</b> . . . . .	103
5.2	Flow chart of <b>lapw2para</b> . . . . .	103
7.1	Form of triangular electric field of 1 Ry for $c = 20$ bohr . . . . .	142
7.2	Schematic dependence of DOS and $u_l(r, E_l)$ on the energy . . . . .	155
9.1	3D electron density in TiC generated with XCrysDen . . . . .	248



## Licence conditions of WIEN2k

P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka, J. Luitz, R. Laskowski, F. Tran and L. D. Marks

Prof. Dr. Peter Blaha  
Vienna University of Technology  
Institute of Materials Chemistry  
Getreidemarkt 9/165-TC  
A-1060 Vienna, Austria  
email: peter.blaha@tuwien.ac.at

### DEFINITIONS:

In the following, the term “the authors”, refers to P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka, J. Luitz, R. Laskowski, F. Tran and L. D. Marks and 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 with the following two citations:
  - ▶ P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka, J. Luitz, R. Laskowski, F. Tran and L. D. Marks, **WIEN2k**, An Augmented Plane Wave + Local Orbitals Program for Calculating Crystal Properties (Karlheinz Schwarz, Vienna University of Technology, Austria), 2018. ISBN 3-9501031-1-2
  - ▶ WIEN2k: An APW+lo program for calculating the properties of solids: P. Blaha, K. Schwarz, F. Tran, R. Laskowski, G. K. H. Madsen, and L. D. Marks (2020), J. Chem. Phys., 152:074101.

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 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 thirty 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-type 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/Linux-type operating system since the programs are linked together via C-shell scripts. It has been implemented successfully on many different Intel (AMD) based computer systems running under Linux, but also on the IBM RS6000 series. It is expected to run on any modern LINUX (UNIX) system.

Hardware requirements will change from case to case (small cases with 60 atoms per unit cell can be run on any Intel (AMD) based PC/Laptop, but generally we recommend a powerful PC or workstation with a modern Intel I7/I9 multi-core cpu and at least 16 GB (better: 4-8 Gb/core or more) memory and 1-2 Tb of disk space. It can utilize multi-core shared memory hardware efficiently up to 4-8 cores. For a very efficient coarse grain parallization on the k-point level, a cluster of PCs with Gb/s network is sufficient. Faster communication (infiniband or a larger multi-core node with modern Intel-Xeons) is needed for the fine grain (single k-point) mpi-parallel version, which is necessary for unit cells with more than 100 atoms.

In order to use the main options (non-mpi) and features (such as the graphical user interface **w2web** or some of its plotting tools) the following (public domain) program packages must be installed:

- ▶ Fortran 90 compiler (we recommend Intels ifort, but gfortran is fine too)
- ▶ efficient Blas/Lapack libraries (Intels mkl, or a recent OpenBlas library)
- ▶ FFTW3
- ▶ csh or tcsh
- ▶ perl 5 or higher (for **w2web** only)
- ▶ an editor of your choice (vi, "xterm -e vi", emacs, nedit, gedit, ...)
- ▶ ghostscript (with jpg support)
- ▶ gnuplot (with png support)
- ▶ www-browser
- ▶ pdf-reader (okular, evince, xpdf, ...)

Usually these packages should be available on any modern Linux system. If one of these packages is not available by default, it can usually be installed easily from the Linux distribution or from public domain sources (see Chapt. 11) or the corresponding configuration may be changed (e.g. using vi instead of emacs). Additional software is needed for the mpi-version (see Chapt. 11) but is only necessary when the corresponding high-end hardware is available AND you want to run cases with at least 50 atoms or more.

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:

- ▶ G. Abo (Univ.Alabama): numerous bug fixes, outstanding contributions to the mailing list
- ▶ C. Ambrosch-Draxl (Univ. Graz, Austria) and her group: optics
- ▶ M. Bagheri (Vienna): pes
- ▶ K. Belbase (Vienna): lcore potential extensioni, stress tensor
- ▶ T. Charpin (Paris): elastic constants
- ▶ J. Doumont (Vienna): LDA-1/2i, diret tau calculation, gKS meta-GGA calculations
- ▶ H. Hofstaetter and O.Koch (Vienna): iterative diagonalization
- ▶ C. Först (Vienna): afminput
- ▶ M. Jamal (Iran): various scripts, 2DRoptimize, IRelast, Tmaker
- ▶ K. Jorissen (Univ.Antwerp), C.Hebert (TU Wien): telnes3
- ▶ E. Kabliman (TU Vienna): arrows
- ▶ L. Kalantari (TU Vienna): l-mBJ
- ▶ F. Karsai (TU Vienna): elast, lapwso, HDLOs
- ▶ W. Lafargue-Dit-Hauret (Rennes): init\_orb.lapw, \*.cf files in qtl
- ▶ R. Luke (Univ. Delaware): new mixer (MSEC1)
- ▶ M. Nelhiebel, P. Schattschneider (Vienna), Kevin Jorissen (Univ. Antwerp), C.Hebert (TUW): (telnes)
- ▶ P. Novák and J. Kuneš (Prague): LDA+U, SO, lapwdm, qtl, dipan
- ▶ P. Ondracka: OpenMP parallelization (lapw0, lapw1, lapw2, optic)
- ▶ C. Persson (Uppsala): irreducible representations
- ▶ O. Rubel (McMasters Univ): BerriPy, mstar
- ▶ T. Ruh (Vienna): ELPA-interface, 3ddens, nlvdw
- ▶ M. Scheffler (Fritz Haber Inst., Berlin): and his group, forces, dstart, geometry optimization
- ▶ E. Sjöstedt and L Nordström (Uppsala, Sweden): APW+lo
- ▶ J. Sofu and J. Fuhr (Barriloché): Bader analysis
- ▶ P. Wissgott, E.Assmann and J.Kunes (TU Vienna): wien2wannier
- ▶ B. Yanchitsky and A. Timoshevskii (Kiev): sgroup

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

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



This work was supported by the following institutions:

- ▶ Austrian Science Foundation (FWF-Projects P5939, P7063, P8176, SFB08-11, SFB-Vicom, DK-Solids for Fun)
- ▶ L.D.Marks was supported by NSF.

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

`peter.blaha@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

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

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

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

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

$$\rho_\sigma(r) = \sum_{i,k} \rho_{ik}^\sigma |\chi_{ik}^\sigma(r)|^2 \quad (2.2)$$

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  $E_{tot}$  gives the Kohn-Sham equations (in Ry atomic units),

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

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

Recent progress has been made going beyond the LSDA by adding gradient terms of the electron density to the exchange-correlation energy or its corresponding potential. This has led to the generalized gradient approximation (GGA) in various parameterizations, e.g. 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], TPSS [Tao et al., 2003] and in particular SCAN [Sun et al., 2015b] and TM [Tao and Mo, 2016] employ for the evaluation of the exchange-correlation energy not only the gradient of the density, but also the kinetic energy density  $\tau(r)$ . Such meta-GGA calculations can be done non-self-consistent (a posteriori from a converged GGA calculation) or self-consistent within the generalized Kohn-Sham (gKS) scheme. Such calculations are not much more expensive than standard GGA and highly recommended for more accurate binding energies.

Band gaps can be calculated efficiently and accurately using the Tran-Blaha modified Becke-Johnson (TB-mBJ) potential ([Tran and Blaha, 2009]). A local version of the TB-mBJ potential [Rauch et al., 2020] has been proposed for interfaces and systems with vacuum.

Van der Waals interactions can be included by using the DFT-D3 [Grimme et al., 2010, Grimme et al., 2011] or DFT-D4 [Caldeweyher et al., 2017, Caldeweyher et al., 2019, Caldeweyher et al., 2020] method of Grimme, or by using a nonlocal van der Waals functional [Dion et al., 2004].

For correlated electron systems fast schemes like DFT+U and EECE ([Tran et al., 2006]) are also implemented.

Last but not least, also Hartree-Fock and in particular (screened) Hybrid-DFT is also available ([Tran and Blaha, 2011]).

## 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.

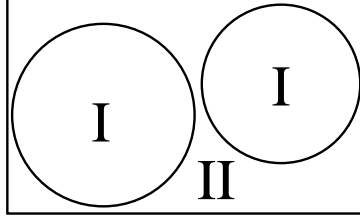


Figure 2.1: Partitioning of the unit cell into atomic spheres (I) and an interstitial region (II)

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

1. (I) inside atomic sphere  $t$ , of radius  $R_t$ , a linear combination of radial functions times spherical harmonics  $Y_{lm}(r)$  is used (we omit the index  $t$  when it is clear from the context)

$$\phi_{\mathbf{k}_n} = \sum_{lm} [A_{lm,\mathbf{k}_n} u_l(r, E_l) + B_{lm,\mathbf{k}_n} \dot{u}_l(r, E_l)] Y_{lm}(\hat{\mathbf{r}}) \quad (2.4)$$

where  $u_l(r, E_l)$  is the (at the origin) regular solution of the radial Schroedinger equation for energy  $E_l$  (chosen normally at the center of the corresponding band with l-like character) and the spherical part of the potential inside sphere  $t$ ;  $\dot{u}_l(r, E_l)$  is the energy derivative of  $u_l$  evaluated at the same energy  $E_l$ . A linear combination of these two functions constitute the linearization of the radial function; the coefficients  $A_{lm}$  and  $B_{lm}$  are functions of  $k_n$  (see below) determined by requiring that this basis function matches (in value and slope) each plane wave (PW) the corresponding basis function of the interstitial region;  $u_l$  and  $\dot{u}_l$  are obtained by numerical integration of the radial Schroedinger equation on a radial mesh inside the sphere.

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

$$\phi_{\mathbf{k}_n} = \frac{1}{\sqrt{\omega}} e^{i\mathbf{k}_n \cdot \mathbf{r}} \quad (2.5)$$

where  $\mathbf{k}_n = \mathbf{k} + \mathbf{K}_n$ ;  $\mathbf{K}_n$  are the reciprocal lattice vectors and  $\mathbf{k}$  is the wave vector inside the first Brillouin zone. Each plane wave is augmented by an atomic-like function in every atomic sphere.

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

$$\psi_{\mathbf{k}} = \sum_n c_n \phi_{\mathbf{k}_n} \quad (2.6)$$

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

In order to improve upon the linearization (i.e. to increase the flexibility of the basis) and to make possible a consistent treatment of semicore and valence states in one energy window (to ensure

orthogonality) additional ( $k_n$  independent) basis functions can be added. They are called “local orbitals (LO)” (Singh 91) and consist of a linear combination of 2 radial functions at 2 different energies (e.g. at the 3s and 4s energy) and one energy derivative (at one of these energies):

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

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:

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

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

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.

In order to remove any larger linearization error, one can add multiple LOs (HELOs at higher energies to improve the unoccupied states several Ry above EF), and HDLOs which contain the second energy derivative of the radial wave function  $\ddot{u}_l(r)$  ([Karsai et al., 2017]).

## 2.2.3 General considerations

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

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

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 computational efficiency and numerical accuracy [Krimmel et al., 1994].

Recently, a formalism for the stress tensor has been presented [Belbase et al., 2021] and implemented in WIEN2k. Unfortunately it is restricted to NREL (non-relativistic) calculations.

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

X-ray absorption and emission and electron energy loss (ELNES) 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]. Supercells allow to include core-holes.

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

Optical properties (dielectric function, optical conductivity, ...) 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-Draxl and Sofo, 2006]. A Kramers-Kronig transformation is also available.

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].

Valence band Photoelectron spectroscopy (XPS, UPS) can be calculated from the partial DOS and the corresponding atomic cross sections [Bagheri and Blaha, 2019].

Hyperfine interactions (Isomer shifts, Hyperfine fields, Quadrupole splittings, NMR shifts in insulators and metals) as measured by Mössbauer, NMR or PAC spectroscopy.

Phonons with an interface to K.Parlinski's PHONON or A. Togos Phonopy program.



---

## 3 Quick Start

---

### Contents

---

3.1	Naming conventions . . . . .	13
3.2	Starting the server . . . . .	14
3.3	Connecting to the <b>w2web</b> server . . . . .	15
3.4	Creating a new session . . . . .	15
3.5	Creating a new case . . . . .	15
3.6	Creating the struct file . . . . .	15
3.7	Initialization . . . . .	18
3.8	The SCF calculation . . . . .	21
3.9	The case.scf file . . . . .	23
3.10	Saving a calculation . . . . .	23
3.11	Calculating properties . . . . .	24
3.12	Setting up a new case . . . . .	32

---

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 to setup and run the calculations using the graphical user interface **w2web** or the commandline interface.

### 3.1 Naming conventions

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

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

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

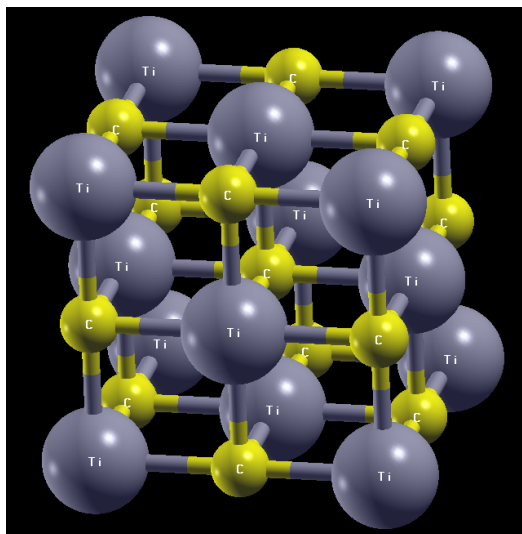


Figure 3.1: TiC in the sodium chloride structure. This plot was generated using XCrysDen (see 9.31.1), which shows up in **w2web** when xcrsden is properly installed. Alternatively, also VESTA (see 9.31.2) can plot WIEN2k-structures.

```
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 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.

The **w2web** server will run until the computer is rebooted. This may cause problems when you logout and login remotely, because the **DISPLAY** variable for **XCrysDen** may no longer be valid. Thus we recommend to kill the server before you logout using `kill w2web` and restart the server when you login again.

### 3.3 Connecting to the w2web server

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

```
firefox 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 \$W2WEB\_CASE.BASEDIR directory if no working directory was designated to this session previously (or if the directory does not exist any more).

### 3.5 Creating a new case-directory

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

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

### 3.6 Creating the “master input” file case.struct

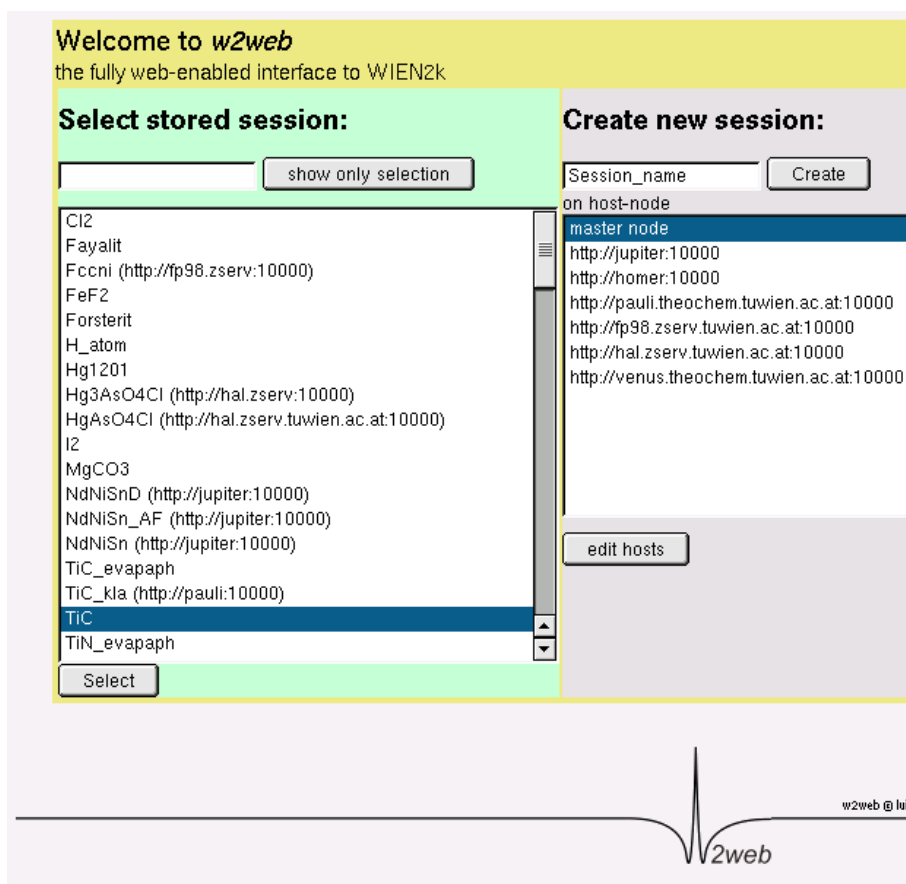
To create the file **TiC.struct** start the struct-file generator using “Execution  StructGen” (see figure 3.4).

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

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

Title	TiC
Lattice	F (for face centered)
a	4.328 Å (make sure the Ang button is selected)
b	4.328 Å
c	4.328 Å
$\alpha, \beta, \gamma$	90
Atom	Ti, enter position (0,0,0)
Atom	C, enter position (.5,.5,.5)

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

Figure 3.2: Startup screen of *w2web*

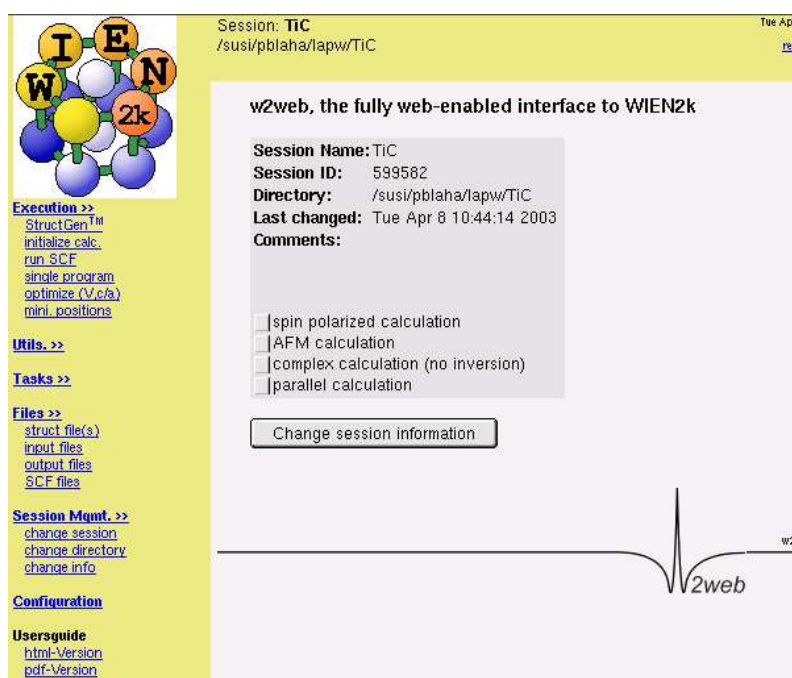


Figure 3.3: Main window of w2web

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) or a volume optimization, which would require reduced RMT values.

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

When you are done, exit **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.

If XCrysDen is installed in your PATH, you can now view the structure using “Execution  view structure”.

A few other hints on **StructGen**:

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

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

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

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

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

### 3.7 Initialization of the calculation (init\_lapw)

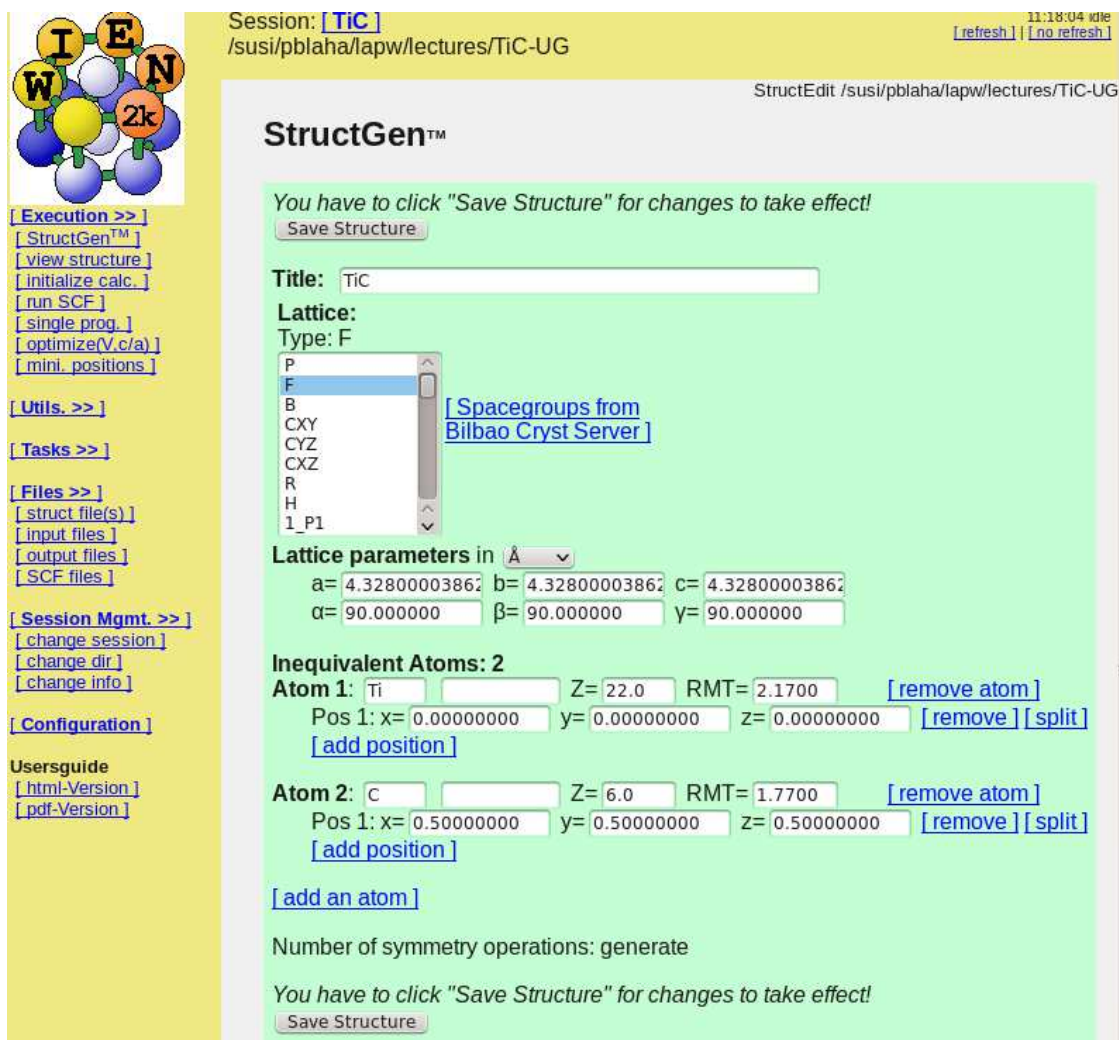
After the basic input file has been created, initialization of the calculation is done by “Execution  initialize calc.” (see figure 3.5). For structures given by experiment (not man-made supercells,...) you would usually run in “Fast mode”, where you can specify the desired precision and a few other parameters and proper input files will be automatically generated (RKmax, GMAX, k-mesh). However, this introduction will guide you through the “Individual mode” necessary to initialize the calculation, so that you get an idea about the different steps. Simply follow the steps that are highlighted in green and follow the instructions.

The initialization process is also described in detail in section 5.1.3.

For a manual initialization you have to run several steps one after the other and check the screen for possible error or warning messages. **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  $f \cdot$  NN-dist. are calculated)

**view TiC.outputnn** : check for overlapping spheres, coordination numbers, nearest neighbor distances and angles, (e.g. in the sodium chloride structure there must be 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 possibly 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. See Sec.4.3.



The screenshot displays the StructGen web interface. At the top, the session title is "TiC" and the URL is "/susi/pblaha/lapw/lectures/TiC-UG". The interface includes a sidebar with navigation links such as "Execution >>", "Utils. >>", "Tasks >>", "Files >>", "Session Mgmt. >>", "Configuration", and "Usersguide". The main content area shows the "StructGen™" logo and a message: "You have to click 'Save Structure' for changes to take effect!". Below this, there is a "Save Structure" button, a "Title" field containing "TiC", and a "Lattice:" section with "Type: F" and a list of space groups including P, F, B, CXY, CYZ, CXZ, R, H, and I\_P1. A link for "Spacegroups from Bilbao Cryst Server" is provided. The "Lattice parameters in Å" section shows a=4.32800003862, b=4.32800003862, c=4.32800003862, and angles alpha=beta=gamma=90.000000. Under "Inequivalent Atoms: 2", two atoms are listed: Atom 1 (Ti) at (0,0,0) with Z=22.0 and RMT=2.1700, and Atom 2 (C) at (0.5,0.5,0.5) with Z=6.0 and RMT=1.7700. Each atom has "remove atom" and "add position" buttons. At the bottom, there is an "add an atom" button and a "Number of symmetry operations: generate" field. A final "Save Structure" button is at the very bottom.

Figure 3.4: StructGen of w2web

Figure 3.5: Initialization with **w2web**

**x sgroup** calculates the point and spacegroups for the given structure

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

**x symmetry** generates from a raw **case.struct** file the space group symmetry operations, determines the point group of the individual atomic sites, generates the LM expansion for the lattice harmonics (in **case.in2\_st**) and local rotation matrices (in **case.struct\_st**).

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

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

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

**edit TiC.outputst** : check the output (did you specify a proper atomic configuration, did lstart converge, are the core electrons confined to the atomic sphere, what are my core and valence and semicore states?). Warnings for the radial mesh can usually be neglected since it affects only the atomic total energy. **lstart** generates **TiC.in0\_st**, **in1\_st**, **in2\_st**, **inc\_st** and **inm.st**. For Ti it selects automatically  $1s$ ,  $2s$ , and  $2p$  as core states, the semicore  $3s$  and  $3p$



will be treated with local orbitals together with  $3d$ ,  $4s$  and  $4p$  valence states.

**edit TiC.in1.st** : As mentioned, in batch mode the input files are generated automatically with some default values which should be a reasonable choice for most cases. Here we 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 5.0-9.0 (3.0 if you have small H-spheres) are usually reasonable, here use 6.6. For more information consult [http://www.wien2k.at/reg\\_user/faq/rkmax.html](http://www.wien2k.at/reg_user/faq/rkmax.html). Here we will just change **EMAX** of the energy window from 1.5 to 2.0 Ry in order to be able to calculate the unoccupied DOS to higher energies.

**edit TiC.in2.st** : Here you may increase the LM expansion, or 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.in1c**, **in2c** are produced.

x **kgen** generates a k-mesh in the Brillouin zone (BZ). You must specify the number of k-points in the whole BZ (use 1000 for comparison with the provided output, a “good” calculation needs at least 10 times as much for such a small unitcell and metallic character). For details see section 6.5.

**view TiC.klist** : check the number of k-points in the irreducible wedge of the BZ (IBZ). 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  $g_{max} > g_{min}$ )

**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 a desired precision or 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 (eg. 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.

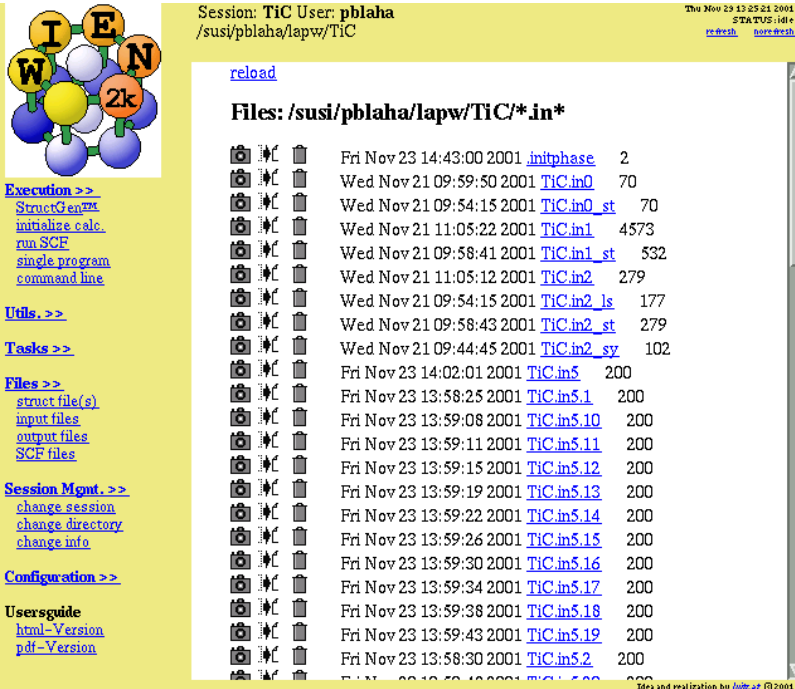
Initialization of a calculation (running **init\_lapw**) will create all inputs for the subsequent SCF calculation choosing some default options and values. You can find a list of input files using “Files [input files](#)” (3.6).

## 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



Session: TIC User: pblaha  
/susi/pblaha/lapw/TiC

Thu Nov 23 13:25:21 2001  
STATUS: idle  
[refresh](#) [logout](#)

[reload](#)

**Files: /susi/pblaha/lapw/TiC/\*.\***

			Fri Nov 23 14:43:00 2001	<a href="#">_initphase</a>	2
			Wed Nov 21 09:59:50 2001	<a href="#">TiC.in0</a>	70
			Wed Nov 21 09:54:15 2001	<a href="#">TiC.in0_st</a>	70
			Wed Nov 21 11:05:22 2001	<a href="#">TiC.in1</a>	4573
			Wed Nov 21 09:58:41 2001	<a href="#">TiC.in1_st</a>	532
			Wed Nov 21 11:05:12 2001	<a href="#">TiC.in2</a>	279
			Wed Nov 21 09:54:15 2001	<a href="#">TiC.in2_ls</a>	177
			Wed Nov 21 09:58:43 2001	<a href="#">TiC.in2_st</a>	279
			Wed Nov 21 09:44:45 2001	<a href="#">TiC.in2_sy</a>	102
			Fri Nov 23 14:02:01 2001	<a href="#">TiC.in5</a>	200
			Fri Nov 23 13:58:25 2001	<a href="#">TiC.in5.1</a>	200
			Fri Nov 23 13:59:08 2001	<a href="#">TiC.in5.10</a>	200
			Fri Nov 23 13:59:11 2001	<a href="#">TiC.in5.11</a>	200
			Fri Nov 23 13:59:15 2001	<a href="#">TiC.in5.12</a>	200
			Fri Nov 23 13:59:19 2001	<a href="#">TiC.in5.13</a>	200
			Fri Nov 23 13:59:22 2001	<a href="#">TiC.in5.14</a>	200
			Fri Nov 23 13:59:26 2001	<a href="#">TiC.in5.15</a>	200
			Fri Nov 23 13:59:30 2001	<a href="#">TiC.in5.16</a>	200
			Fri Nov 23 13:59:34 2001	<a href="#">TiC.in5.17</a>	200
			Fri Nov 23 13:59:38 2001	<a href="#">TiC.in5.18</a>	200
			Fri Nov 23 13:59:43 2001	<a href="#">TiC.in5.19</a>	200
			Fri Nov 23 13:58:30 2001	<a href="#">TiC.in5.2</a>	200

Idea and realization by [Lutz J.](#) © 2001

**Execution >>**  
[StructGen™](#)  
[initialize calc.](#)  
[run SCF](#)  
[single program](#)  
[command line](#)

**Utils. >>**

**Tasks >>**

**Files >>**  
[struct file\(s\)](#)  
[input files](#)  
[output files](#)  
[SCF files](#)

**Session Mgmt. >>**  
[change session](#)  
[change directory](#)  
[change info](#)

**Configuration >>**

**Usersguide**  
[html-Version](#)  
[pdf-Version](#)

Figure 3.6: List of input files

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 (9 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:

- ▶ select a higher precision (2) and “nodstart” in “Execution  Initialize calc.” with “Fast mode”.
- ▶ or increase RKMAX and GMAX in **case.in1** and **case.in2**
- ▶ or increase the k-mesh with **x kgen** (“Execution  single program”)
- ▶ or 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 3.7):

The screenshot shows the w2web interface for calculating electron density plots. The interface is divided into a sidebar on the left and a main content area on the right. The sidebar contains several navigation menus: Execution >>, StructGen™, view structure, initialize calc., run SCF, single prog., optimize(V c/a.), mini\_positions, Utils >>, Tasks (with sub-items: El. Dens., DOS, XSPEC, TELNES3, OPTIC, Bandstructure), Files >> (with sub-items: struct file(s), input files, output files, SCF files), Session Mgmt >> (with sub-items: change session, change dir, change info), Configuration, and Usersguide (with sub-items: html-Version, pdf-Version). The main content area is titled "Electron density plots" and contains the following text and buttons:

Session: **TiC-UG2** /susi/pblaha/lapw/lectures/TiC-UG2 17:04:51 [refresh] [no refresh]

**Electron density plots**

You must have a valid TiC-UG2.vector file (from an scf calculation). If you don't have it, you must run "x lapw1" with an appropriate input.

Select E-range for lapw2 for a density without semicore or within an E-window. For proper values check energy-parameters and eigenvalues or band-ranges in the corresponding scf-files

view TiC-UG2.scf1

view TiC-UG2.scf2

x lapw2 Calculate cimval with Emin  and Emax   so

**For difference densities only !**

default valence states:  non-default valence states: put P for all your states

x lstart -sigma Calculate atomic valence densities x lstart Calculate atomic valence densities as defined above

Calculate density with XCrysDen (or create TiC-UG2.in5 / execute lapw5 below)

edit TiC-UG2.in5 Edit input-file

x lapw5 Calculate density

View density with XCrysDen

rhoplots Plot Density or download [TiC-UG2.rho](#) for plotting with your own plotting program

Idea and realization by [Hafner et al.](#) © 2001-2006

Figure 3.7: Task “Electron Density Plots”

- ▶ 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 -emin -1.0**). 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 3.8.

- 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
ORTHO
```

For a detailed description of input options consult section 8.14.3

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

Compare the result with the electron density plotted in the (100) plane (see figure 3.9). The program **gnuplot** (public domain) must be installed on your computer. For more advanced graphics use your favorite plotting package or specify other options in **gnuplot** (see **rhoplot.lapw** how **gnuplot** is called).

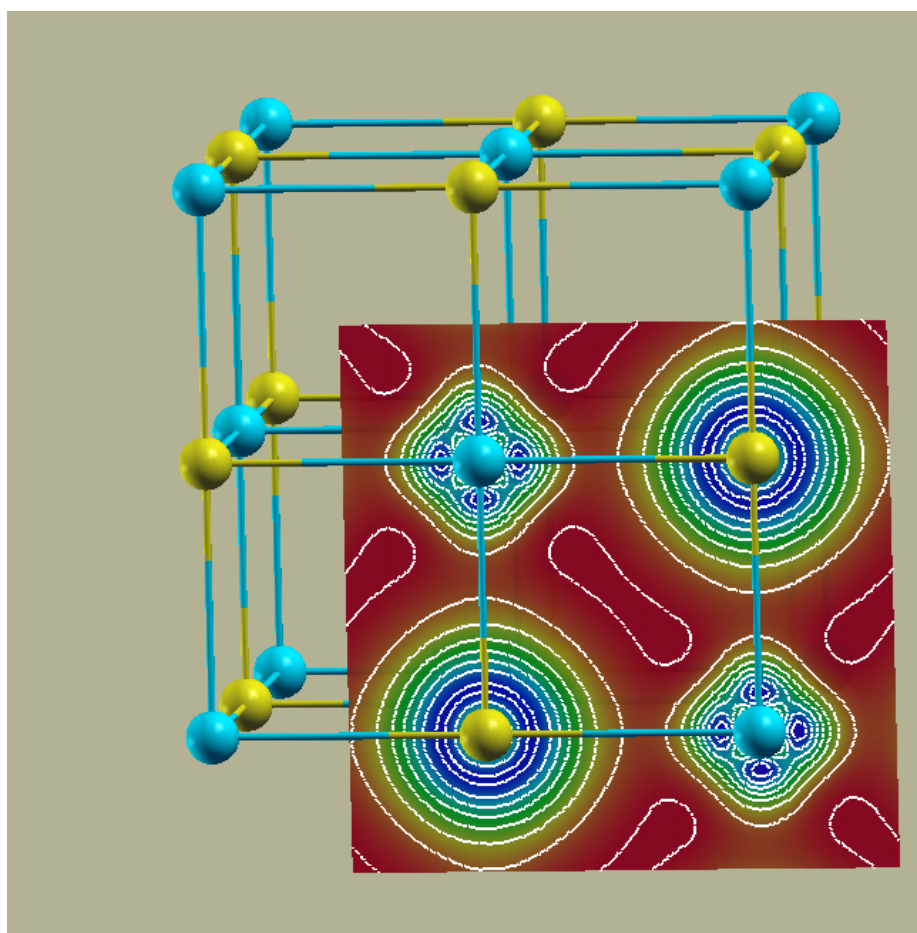


Figure 3.8: Electron density of TiC in (100) plane using Xcrysden

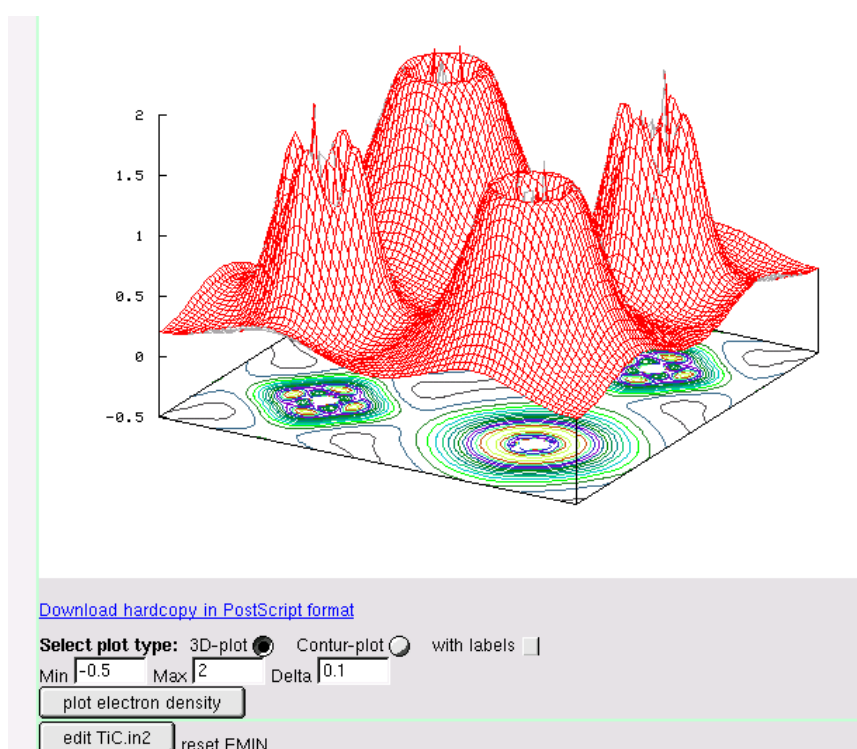


Figure 3.9: Electron density of TiC in (100) plane

### 3.11.2 Density of States (DOS)

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

- ▶ The optional steps allow to select a larger energy window (unoccupied states or a denser k-mesh).
- ▶ Calculate partial charges (“qt1’s”) using **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<sub>eg</sub>, Ti-d<sub>t2g</sub>, C-s and C-p-like DOS.

```

TiC
-0.50      0.00200   1.500 0.003   EMIN, DE, EMAX, Gauss-broadening
6
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 8.27.3

- ▶ Calculate DOS (**x tetra**).
- ▶ Preview output using “dosplot”

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

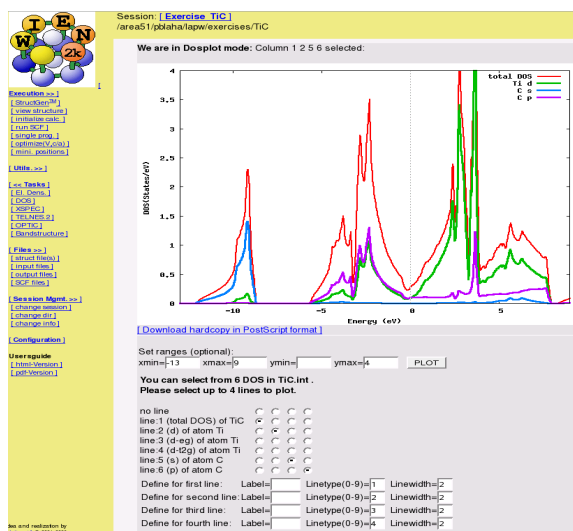


Figure 3.10: Density of states of TiC

The calculated DOS can be compared with figures 3.10 and 3.11. Together with the electron density the partial DOS allows you to analyse the chemical bonding (covalency between  $Ti - d_{eg}$  and  $C - p$ , non-bonding  $Ti - d_{t2g}$ , charge transfer estimates,...)



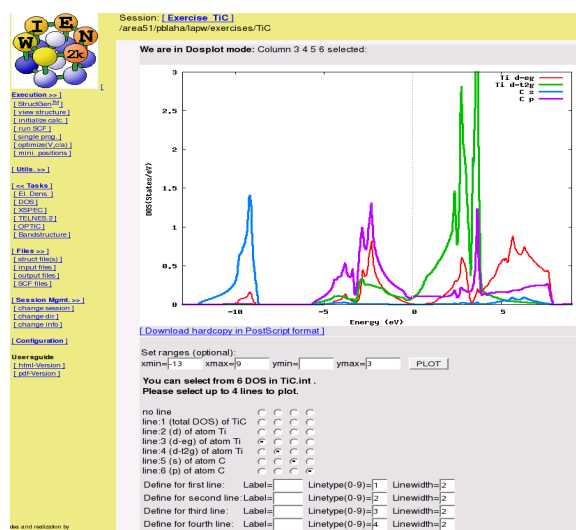


Figure 3.11: Density of states of TiC

### 3.11.3 X-ray spectra

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

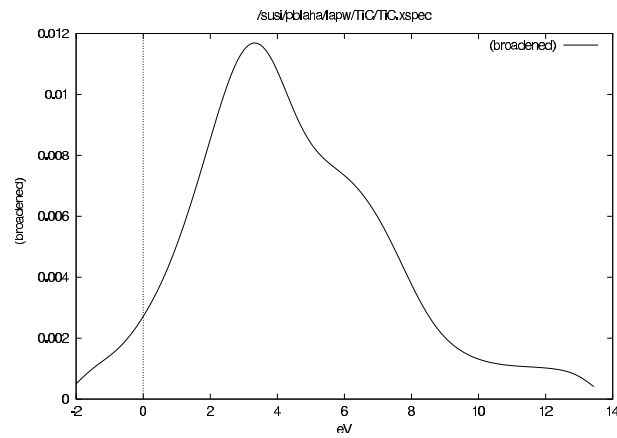
- ▶ Calculate partial charges (**x lapw2 -qt1**). This is only possible, when you still have a valid **TiC.vector** file in a tetrahedral mesh. *To reproduce this figure you will have to increase the EMAX value in your **TiC.in1** to 2.5 Ry and rerun **x lapw1***
- ▶ Edit **TiC.inxs**; choose the offered template. This template will calculate the  $L_{III}$ -spectrum of the first atom (Ti in this example) in the energy range between -2 and 15 eV. For a detailed description of the contents of this input file refer to section 8.28.3.
- ▶ Calculate spectra
- ▶ Preview spectra

If you want to use the supplied plotting interface **speplot** to preview the results, the public domain program **gnuplot** must be installed on your computer. The calculated TiC Ti- $L_{III}$ -spectrum can be compared with figure 3.12.

### 3.11.4 Bandstructure

Select “Bandstructure” from the “Tasks” menu and click on the buttons one by one:

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

Figure 3.12: Ti  $L_{III}$  spectrum of TiC

because the  $k$ -mesh for the band structure plots is *not* suitable for calculations of such properties.

- ▶ Edit **TiC.insp**: insert the correct Fermi energy (which can be found in the saved **scf**-file) and specify plotting parameters. For comparison with figure 3.13 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 3.13.

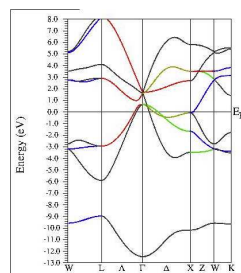


Figure 3.13: Bandstructure of TiC

### 3.11.5 Bandstructure with band character plotting / full lines

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

- ▶ Calculate partial charges (`x lapw2 -qt1 -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 3.14) 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.qt1*` 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.qt1` files.

You can compare your results with figure 3.14.

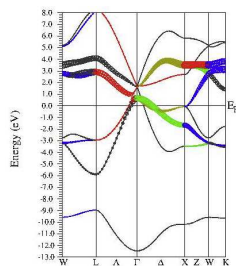


Figure 3.14: Bandstructure of TiC, showing t<sub>2g</sub>-character bands of Ti in character plotting mode

### 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 3.15. 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**.

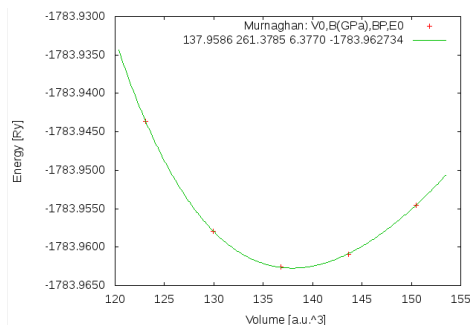


Figure 3.15: Energy vs. volume curve for TiC

## 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 when you know the spacegroup only the inequivalent positions needs to be given. 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

#### Create `case.struct`

Create a new directory **case** in a desired place (do not use your \$HOME-directory, but eg. `~/WIEN2k`, or `~/lapw` ... and change into this directory.

A new `case.struct` file is never 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`”, “`POSCAR`” or “`xyz`” file into the WIEN2k `case.struct` file. Check page 249 for more info on the specific file formats.

You can also use the `struct` file from a similar case (eg. TiN from TiC) 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). For AFM cases generate `case_new.inst` using `instgen_lapwi -ask`.

### Initialize the calculation

Run the script `init_lapw -b [-prec XX]`, which will setup adequate input parameters for the desired precision (-prec 1 is the default).

### Run scf calculation

`run_lapw [-cc 0.0001 -ec 0.00001 ...]` performs an scf cycle until the desired convergence has been reached. You can check the convergence using:

```
grep :DIS case.scf    or    grep :ENE case.scf
```

and view the `case.dayfile` file. In case the scf stops after 40 cycles without convergence or you want to converge with better criteria, simply rerun the `run_lapw` command but add the -NI flag.

All actions of this script are logged and appended in short in `:log` and for the last command in detail in the file `case.dayfile`.

### Save a calculation

Once the scf has finished and you are happy with the convergence, always save the calculation with a meaningful name:

```
save_lapw pbe_prec-1_exp-vol
```

After doing some other calculations (precision, different lattice parameters, ...) you can always come back using :

```
restore_lapw pbe_prec-1_exp-vol
```

## 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 249 for more info on the specific file formats.

For more information on the **StructGen** refer to page 246.

## **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 . . . . .	37
4.2	Input/Output files . . . . .	41
4.3	The case.struct.file . . . . .	42
4.4	The case.scf file . . . . .	46
4.5	Flow of programs . . . . .	48

---

(for naming conventions see section 3.1)

## 4.1 Flow of input and output files

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

```
programX programX.def
```

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

```
x program -d
```

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

**case.struct** a “master” input file, which is described below (Section 4.3)

**case.inX** a specific input file, where X labels the program (see def-files for each program in chapter 6).

**case.outputX** an output file

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

**case.scfX** a file containing only the most significant output (see description below).

**program.error** error report file, should be empty after successful completion of a program (see chapter 6)

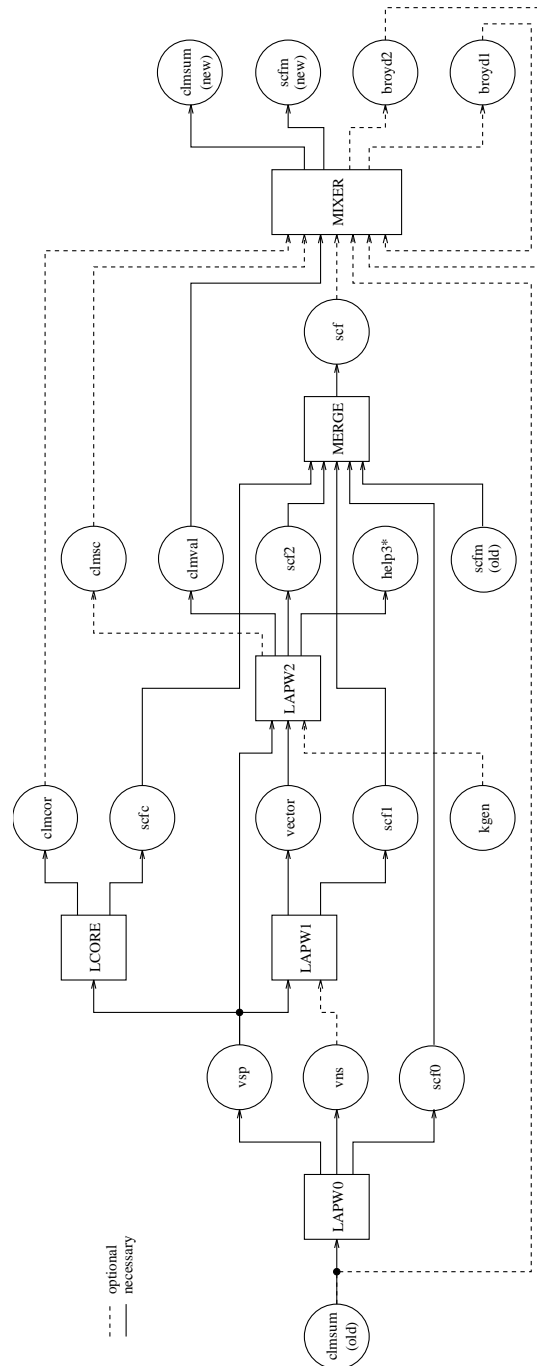


Figure 4.1: Data flow during a SCF cycle (programX.def, case.struct, case.inX, case.outputX and optional files are omitted)

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

program	needs		generates	
	necessary	optional	necessary	optional
NN	nn.def case.struct		case.outputnn	case.struct_nn
SGROUP	case.struct		case.outputsgroup	case.struct_sgroup
SYMMETRY	symmetry.def case.struct	case.in2_st	case.outputs case.in2_st	case.struct_st
LSTART	lstart.def case.struct case.inst		case.outputst case.rsp case.in0_st case.in1_st case.in2_st case.inc_st case.inm_st case.inm_restart	case.rspup/dn case.rsigma case.vsp_st case.vspd_n.st case.sigma case.potup/dn case.sptup/dn case.tspup/dn
KGEN	kgen.def case.struct	case.ksym	case.outputkgen case.klist case.kgen	
DSTART	dstart.def case.struct case.rsp(up) case.in0 case.in1 case.in2	case.indstart case.tsp(up) case.inpd	case.outputd case.clmsum(up) dstart.error case.in0_std	new_super.clmsum(up) case.clmsc(up) case.r2v_half(dn) case.tausum(up)

Table 4.1: Input and output files of init programs

program	needs		generates	
	necessary	optional	necessary	optional
SPAGHETTI	spaghetti.def case.insp case.struct case.output1	case.qtl case.outputso case.irrep	case.spaghetti_ps case.outputsp case.band.agr	case.spaghetti_ene
TETRA	tetra.def case.int case.kgen	case.qtl case.energy case.scf2	case.outputt case.dos1(2,3) case.dos1ev(1,2,3)	
LAPW3	lapw3.def case.struct case.in2 case.clmsum		case.output3 case.rho  case.clmsum	
LAPW5	lapw5.def case.struct case.in5 case.clmval	case.sigma	case.output5 case.rho	case.rho.oned
XSPEC	xspec.def case.inc case.int case.vsp case.struct case.qtl		case.outputx case.dos1ev case.xspec case.txspec case.m1 case.m2	case.coredens
OPTIC	optic.def case.struct case.mat_diag case.inop case.vsp case.vector		case.outputop case.symmat	case.symmat1 case.symmat2
JOINT	joint.def case.injoint case.struct case.kgen case.weight		case.outputjoint case.joint	case.sigma_intra case.intra

continued on next page

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

Table 4.2: Input and output files of utility programs

program	needs		generates	
	necessary	optional	necessary	optional
LAPW0	lapw0.def case.struct case.in0(*) case.clmsum	case.clmup/dn case.vrespsum/up/dn case.inm case.r2v_half(dn) case.tausum/up/dn case.grr case.in0_loc.vsp case.clmcor(up) case.taucor(up)	case.output0 case.scf0 case.vsp(up/dn) case.vns(up/dn)	case.r2v case.vcoul case.vtotal case.ececup/dn case.vtau(up) case.vspmgga
ORB	orb.def case.struct case.inorb case.dmat case.vsp	case.energy case.vorb_old	case.outputorb case.scforb case.vorb orb.error	case.br1orb case.br2orb
LAPW1	lapw1.def case.struct case.in1 case.vsp case.klist	case.vns case.vorb case.vector.old case.vtau case.vspmgga	case.output1 case.scf1 case.vector case.energy	case.nsh(s) case.nmat_only
LAPWSO	lapwso.def case.struct case.inso case.in1 case.vector case.vsp case.vns case.energy	case.vorb	case.vectorso case.outputso case.scfso case.energyso case.normso	
LAPW2	lapw2.def case.struct case.in2 case.vector case.vsp case.energy	case.kgen case.nsh case.weight case.recprlist	case.output2 case.scf2 case.clmval	case.qtl case.weight case.help03* case.vrespval case.almb1m case.radwf case.dmatup/dn case.tauval
LAPWDM	lapwdm.def case.struct case.indm case.vector case.vsp case.weight case.energy	case.inso	case.outputdm case.scfdm case.dmat lapwdm.error	
SUMPARA	case.struct case.clmval	case.scf2p	case.outputsum case.clmval case.scf2	

continued on next page

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

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.almb1m** Contains the  $A_{lm}, B_{lm}, C_{lm}$  coefficients of the wavefunctions (generated optional by **lapw2**).

**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 8.14). 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 8.14). 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.dosrnXev** Contains the renormalized PDOS (without interstitial) generated by **x pes** for plotting with **dosplot2 -ren**

**case.energy** Contains the eigenvalues (in Ry) 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 7.9.3) 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.pesX** Contains the photoelectron spectra from **pes**
- 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 (wave function) on a grid in a specified plane generated by **lapw5** (**lapw7**). This file can be used as input for your favorite contour or 3D plotting program.
- case.rsp** Contains the atomic densities generated by **lstart**. They are used by **dstart** to generate a first crystalline density (**case.clmsum**).
- case.r2v** Contains the exchange potential (in the lattice harmonics representation as  $r^2 * V_{LM}(r)$  and as Fourier coefficients) in a form suitable for plotting with **lapw5**.
- case.scf\_mini** Contains the last scf-iteration of each individual time (geometry) step during a structural minimization using **mini**. Thus this file contains a complete history of properties (energy, forces, positions) during a structural minimization.
- case.sigma** Contains the atomic densities for those states with a “P” in **case.inst**. Generated in **lstart** and used for difference densities in **lapw5**.
- case.spaghetti\_ps** A ps file with the energy bandstructure plot generated by **spaghetti**.
- case.sptup/dn** Contains  $r$  and  $V$  ( $V * r, V * r^2$ ) from **lstart** for plotting potentials of free atoms.
- case.tausum/up/dn** Contains the crystalline kinetic energy densities (slater form  $-\nabla\psi^* \cdot \nabla\psi$ )
- case.tspup/dn** Contains the atomic kinetic energy densities (slater form  $-\nabla\psi^* \cdot \nabla\psi$ )
- case.symmat** Contains the momentum matrix elements between bands  $i,j$ . Created by **optic** and used in **joint**.
- case.vcoul** Contains the Coulomb potential (Ry) (in the lattice harmonics representation as  $r^2 * V_{LM}(r)$  and as Fourier coefficients) in a form suitable for plotting with **lapw5**.
- case.vorb** Contains the orbital potential (in Ry) generated by **orb** for LDA+U or onsite-hybrid-DFT calculations in form of a (2l+1,2l+1) matrix.
- case.vtotal** Contains the total potential (Ry) (in the lattice harmonics representation as  $r^2 * V_{LM}(r)$  and as Fourier coefficients) in a form suitable for plotting with **lapw5**.
- case.vector** Binary file, contains the eigenvalues and eigenvectors of all k-points calculated in **lapw1**. In spin-polarized calculations two files **case.vectorup** and **case.vectordn** are used instead. **lapwso** generates **case.vectorso**.
- case.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 (bohr.Ry) stored as  $r * V$  (thus the first values should be close to  $-2 * Z$ ). In spin-polarized calculations two files **case.vspup** and **case.vspdn** are used instead.
- case.vspmgga** Like above, but it contains the spherical part of the total potential for a gKS MGGA potential.

### 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, POSCAR 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<sub>2</sub> (adding line numbers):

```

----- top of file -----line #
Titaniumdioxide TiO2 (rutile): u=0.305 1
P LATTICE,NONEQUIV. ATOMS 2 2
MODE OF CALC=RELA 3
8.6817500 8.6817500 5.5916100 90. 90. 90. 4
ATOM -1: X= 0.0000000 Y= 0.0000000 Z= 0.0000000 5
MULT= 2 ISPLIT= 8 6
ATOM -1: X= 0.5000000 Y= 0.5000000 Z= 0.5000000
Titanium NPT= 781 R0=.000022391 RMT=2.00000000 Z:22.0 7
LOCAL ROT MATRIX: -.7071068 0.7071068 0.0000000 8
0.7071068 0.7071068 0.0000000 9
0.0000000 0.0000000 1.0000000 10
ATOM -2: X= 0.3050000 Y= 0.3050000 Z= 0.0000000
MULT= 4 ISPLIT= 8
ATOM -2: X= 0.6950000 Y= 0.6950000 Z= 0.0000000
ATOM -2: X= 0.8050000 Y= 0.1950000 Z= 0.5000000
ATOM -2: X= 0.1950000 Y= 0.8050000 Z= 0.5000000
Oxygen NPT= 781 R0=.000017913 RMT=1.60000000 Z: 8.0
LOCAL ROT MATRIX: 0.0000000 -.7071068 0.7071068
0.0000000 0.7071068 0.7071068
1.0000000 0.0000000 0.0000000

16 SYMMETRY OPERATIONS: 11
1 0 0 0.00 12
0 1 0 0.00 13
0 0 1 0.00 14
1 15
1 0 0 0.00
0 1 0 0.00
0 0 -1 0.00
2
.....
0 1 0 0.50
-1 0 0 0.50
0 0 1 0.50
16
----- bottom of file -----

```

Interpretive comments on this file are as follows.

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

Table 4.4: Lattice type, description and bravais matrix used in **WIEN2k**. The angle  $\gamma'$  is defined via  $\cos(\gamma) = \cos(\gamma') \sin(\alpha) \sin(\beta) + \cos(\beta) \cos(\alpha)$

**line 1:** format (A80)  
title (compound)

**line 2:** format (A4,23X,I3)  
lattice type, NAT

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. Possibly 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 3:** format (13X,A4)  
mode

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

**line 4:** format (6F10.6)  
a, b, c,  $\alpha$ ,  $\beta$ ,  $\gamma$

a, b, c unit cell parameters (in a.u., 1 a.u. = 0.529177 Å). In face- or body-centered structures the non-primitive (cubic) lattice constant, for rhombohedral (R) lattices the hexagonal lattice constants must be specified. (The following may help you to convert between hexagonal and rhombohedral specifications:  

$$a_{hex} = 2\cos\left(\frac{\pi - \alpha_{rhombo}}{2}\right)a_{rhombo}$$

$$c_{hex} = 3\sqrt{a_{rhombo}^2 - \frac{1}{3}a_{hex}^2}$$
and (for fcc-like lattices)  $a_{rhombo} = a_{cubic}/\sqrt{2}$

$\alpha, \beta, \gamma$  angles between unit axis (if omitted, 90° is set as default). Set it only for P and CXZ lattices

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

atom-index running index for inequivalent atoms

positive in case of cubic symmetry  
negative for non-cubic symmetry  
this is set automatically using **symmetry**

x,y,z position of atom in internal units, i.e. as **positive** fractions of unit cell parameters. ( $0 \leq x \leq 1$ ; the positions in the unit cell are consistent with the convention used in the International Tables of Crystallography [Tab, 1964]. In face- (body-) centered structures only one of four (two) atoms must be given, eg. in Fm3m position 8c is specified with 0.25, 0.25, 0.25 and .75, 0.75, 0.75). For R lattice use rhombohedral coordinates. (To convert from hexagonal into rhombohedral coordinates use the auxiliary program **hex2rhombo**, which can be called at a command-line:

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

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

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



multiplicity	number of equivalent atoms of this kind
isplit	this is just an output-option and is used to specify the decomposition of the lm-like charges into irreducible representations, useful for interpretation in case.qtl). This parameter is automatically set by <b>symmetry</b> :
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,d-yz) e.g.:hcp
4	combining option 1 and 3 e.g.:hcp
5	all d symmetries separate
6	all p symmetries separate
8	combining option 5 and 6
-2	d-z2, d-x2y2, d-xy, (d-xz,d-yz)
88	split <i>lm</i> like charges (for old <b>telnes</b> , not necessary anymore)
99	calculate cross-terms (for old <b>telnes</b> , 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; <i>always use an odd number of mesh points!</i> ) the radial mesh is given on a logarithmic scale: $r(n) = R_0 * e^{[(n-1)*DX]}$
R0	first radial mesh point (typically between 0.0005 and 0.00005, smaller for heavy elements, bigger for light ones; a struct-file generated by w2web will have proper R0 values.)
RMT	atomic sphere radius (muffin-tin radius), can easily be estimated after running <b>nn</b> (see 6.1) and are set automatically with <b>set rmt.lapw</b> see 5.2.5). 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 <b>case.in1</b> .
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 7.9.3 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

**line 16:** free format (optional)

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

<b>:ENE</b>	total energy (Ry). If there is a "WARNING" mentioned, check <b>:WAR</b>
<b>:WAR</b>	contains some warnings indicating that there might be a problem with your calculations. Usually these problems are not fatal, but may influence the accuracy. You should check the calculation when the warning also appears in the last cycle.
<b>:INFO</b>	indicates a "unusual event", that the code has encountered. Usually only for information.
<b>:DIS</b>	charge distance (in $e^-$ ) between last 2 iterations ( $\int  \rho_n - \rho_{n-1}  dr$ ). Good convergence criterium.
<b>:STRESS_GPa00x</b> (x=1,3)	Stresstensor in GPa, only complete, if "total" is printed on the right side.
<b>:FER</b>	Fermi energy (in Ry at internal E-scale) and Fermi-method
<b>:GAP</b>	energy gap (for insulators). Please note, this value will only be correct, if the VBM/CBM are in your k-mesh. (Coarse "shifted" k-meshes do not contain the Gamma-point and the quoted gap might be significantly larger than the real one !!)
<b>:FORxxx</b>	force on atom xxx in mRy/bohr (in the local (for each atom) cartesian coordinate system)
<b>:FGLxxx</b>	force on atom xxx in mRy/bohr (in the global coordinate system of the unit cell (in the same way as the atomic positions are specified, depends on the lattice. Only complete when "total" is printed on the right side.))
<b>:FR</b>	in MSR1a mode prints information about the remaining size of the forces and whether it will/has switched to MSR1 mode.
<b>:APOSxxx</b>	atomic positions and their changes during MSR1a optimization
<b>:DToxx</b>	total difference charge density for atom xx between last 2 iterations
<b>:CTOxx</b>	total charge (in $e^-$ ) in sphere xx (mixed after MIXER)
<b>:NTOxx</b>	total charge in sphere xx (new (not mixed) from LAPW2+LCORE)
<b>:QTLxx</b>	partial charges in sphere xx
<b>:EPLxx</b>	l-like partial charges and "mean energies" in lower (semicore) energy window for atom xx. Used as energy parameters in <code>case.in1</code> for next iteration
<b>:EPHxx</b>	l-like partial charges and "mean energies" in higher (valence) energy window for atom xx. Used as energy parameters in <code>case.in1</code> for next iteration
<b>:EIG</b>	eigenvalues (Ry) of first k-point
<b>:BAN</b>	band ranges (emin - emax of a certain band, Ry) and occupation (electrons)

:EFGxx	Electric field gradient (EFG) $V_{zz}$ for atom xx (related to the quadrupole splitting measured by NMR or Mössbauer spectroscopy)
:ETAxx	Asymmetry parameter of EFG for atom xx
:RTOxx	Density for atom xx at the nucleus (at first radial mesh point, related to Mössbauer Isomer shifts)
:VZERO	Gives the total, Coulomb and xc-potential (Ry) at $z=0$ and $z=0.5$ (meaningfull only for ithe workfunction (VZERO - EF) in slab calculations)
:1S xxx:	1s core eigenvalue of atom xxx. Similar labels for other core states.

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 (bohr)
:VOL	Volume of the unit cell
:IFFT	FFT mesh and enhncement factor for xc-potential/energy in interstitial
:POSxx	Atomic positions for atom xx (as in <b>case.struct</b> )
:RKM	Actual matrix size and resulting RKmax
:LMAX-WF	max angular momentum for spherical and nonspherical Hamilton matrix elements inside sphere
:KPT	number of k-points in IBZ
:GMA	GMAX of density and potential Fourier expansion
:CINT	Core charge, should be very close to an integer (except for core-hole calculations)
: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
:MMIxxx	Spin magnetic moment of atom xxx. Note, that this value depends on RMT.
:CUPxx	spin-up charge (mixed) in sphere xx
:CDNxx	spin-dn charge (mixed) in sphere xx
:NUPxx	spin-up charge (new, from lapw2+lcore) in sphere xx
:NDNxx	spin-dn charge (new, from lapw2+lcore) in sphere xx
:ORBxx	Orbital magnetic moment of atom xx (needs SO calculations and LAPWDM).
:HFFxx	Hyperfine field of atom xx (in kGauss).

During an scf cycle you would mainly monitor convergence parameters like :ENE, :DIS, :FGL, :FR (for MSR1a optimization of atomic positions). You can also use the **scfmonitor\_lapw** script to monitor these quantities graphically, see sec.5.2.10.

If a calculation crashes, :WAR may give some hints, but also check for sudden changes or large oszillations in :NTO, :CTO, :FER or :BAN.

In spin-polarized calculations :MMT and :MMI are crucial quantities which could be used as convergence criterion.

If you are interested on a certain property (:GAP, :EFG, :HFF, :VZERO, ...) monitor it in particular to be sure that they are converged.

It is always best to monitor several quantities, because often one quantity is converged, while another still changes from iteration to iteration. The script **run\_lapw** has three different convergence criteria built in, namely the total energy, the atomic forces and the charge distance (see 5.1.3, 5.1.4).

We recommend the use of UNIX commands like :

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

for monitoring such quantities.

You may define an **alias** for convenience (eg. for `grep -i`, so that your search is not case sensitive; see sec. 11.2).

We provide also a csh-script **grepline\_lapw** to get a quantity from several scf-files simultaneously (sec. 5.2.9 and 5.3).

## 4.5 Flow of programs

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

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

The initialization consists of running a series of small auxiliary programs, which generates the inputs for the main programs. One starts in the respective **case/** subdirectory and defines the structure in **case.struct** (see 4.3). The initialization can be invoked by the script **init\_lapw** (see sec. 3.7 and 5.1.3), and consists of running:

- 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.

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

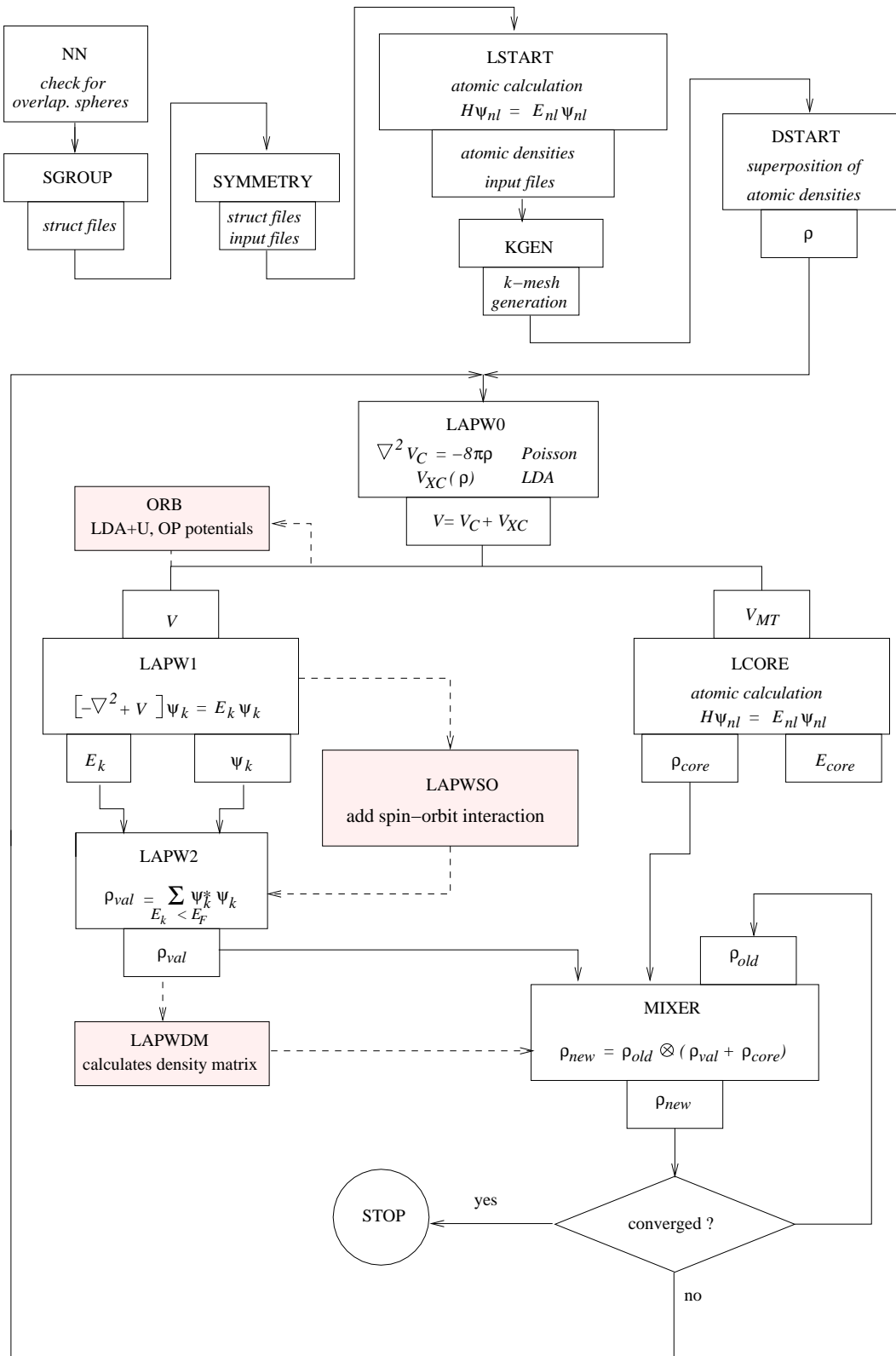


Figure 4.2: Program flow in WIEN2k

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.

The FSM method can also be useful to calculate the ground state magnetic moment of a particular energy functional (eg. a MGGA) in a non self-consistent way. In such a case, a GGA potential  $v_{xc}$  like PBE has to be chosen for the self-consistent calculations, but the total energy is evaluated for the various selected moments. This may introduce some error, but probably very small in most cases.

Please note, that once **runfsm\_lapw** has finished, only **case.vectordn** is ok, but **case.vectorup** is NOT the proper up-spin vector and MUST NOT be used for the calculations of QTLs (and DOS). It must be regenerated by **x lapw1 -up** (see also the comments for iterative diagonalization in section 5.2.22).

### 4.5.4 Staggered field inside atomic spheres to vary the magnetic moment

A simple way to increase or decrease the magnetic moment is to apply a staggered field. The spin-up and spin-down potentials in the Kohn-Sham equations are shifted in opposite directions. For instance, in the case of antiferromagnetism, adding negative and positive shifts inside the atomic spheres for the majority and minority spin electrons, respectively, will increase the magnetic moment around the atoms.

This method is especially useful to study the magnetism with functionals, which are not implemented self-consistently, by searching for the magnetic configuration or magnitude of the moments

that leads to the most negative total energy. This would be approximately equivalent to a self-consistent calculation with the corresponding MGGA potential. As in the case of the FSM method (section 4.5.3), a GGA potential like PBE has to be chosen for the self-consistent calculations.

To turn on the staggered field, the keyword "STAGFIELD" has to be specified at the beginning of the 4th line in **case.in0**. The atoms and the value of the shift (typically in the range 0.01-0.1 Ry) are specified at the 5th and following lines in **case.in0** (see section 7.1.3 for details). The specified shift is the one added to the spin-up potential (the same shift but with opposite sign is applied to the spin-down potential). No shift can be applied in the interstitial region.

### 4.5.5 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. 9.3).
  - ▶ 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). Possibly the system may not want to be antiferromagnetic (but for instance it is ferrimagnetic!).

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

### 4.5.6 Spin-orbit interaction

You can add spin-orbit interaction in LAPWSO (called directly after LAPW1) using a 2nd 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 **init\_so\_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. **init\_so\_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 **init\_so\_lapw**) sorts these operations in the proper way.
- ▶ In a spin-polarized case without inversion symmetry in the original structure, you must not “add inversion” in KGEN and should create the k-mesh using **x kgen -so**.

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**
- ▶ **init\_so\_lapw**
- ▶ **run[sp]\_lapw -so**

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

### 4.5.7 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**, or more conveniently, use **init\_orb\_lapw -orb**, but still you must modify the templates 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\_c\_lapw**.

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

It is also possible to combine an LDA+U potential with the effect of an external magnetic field. In this case, you need two input files, **case.inorb.Bext** and **case.inorb.U**, which are identical to the usual **case.inorb** for external fields and LDA+U, respectively (see section 7.5.3). You can run this using:

► `runsp_lapw -orbext [-so]`

#### 4.5.8 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.4.5.9) or the mBJ potential (see Sec.4.5.11)

##### Onsite hybrid functionals for correlated electrons:

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

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

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

- **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**
- **SCAN**: XC\_SCAN 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.ineec**.

$$E_{xc}^{\text{onsite-B3PW91}}[\rho] = E_{xc}^{\text{LDA}}[\rho] + 0.2 (E_x^{\text{HF}}[\Psi_{\text{corr}}] - E_x^{\text{LDA}}[\rho_{\text{corr}}]) \\ + 0.72 (E_x^{\text{B88}}[\rho] - E_x^{\text{LDA}}[\rho]) \\ + 0.81 (E_c^{\text{PW91}}[\rho] - E_c^{\text{LDA}}[\rho])$$

where  $E_c^{\text{LDA}} = E_c^{\text{PW92}}$ .

- **B3LYP:** XC\_B3LYP in **case.in0**. mode = HYBR and fraction = 0.2 in **case.ineec**.

$$E_{xc}^{\text{onsite-B3LYP}}[\rho] = E_{xc}^{\text{LDA}}[\rho] + 0.2 (E_x^{\text{HF}}[\Psi_{\text{corr}}] - E_x^{\text{LDA}}[\rho_{\text{corr}}]) \\ + 0.72 (E_x^{\text{B88}}[\rho] - E_x^{\text{LDA}}[\rho]) \\ + 0.81 (E_c^{\text{LYP}}[\rho] - E_c^{\text{LDA}}[\rho])$$

where  $E_c^{\text{LDA}} = E_c^{\text{VWN5}}$ .

### Onsite exact-exchange functionals for correlated electrons:

Onsite Hartree-Fock calculations, i.e.,

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

are also possible with the following semilocal functionals.

- **LDA:** XC\_LDA in **case.in0**. mode=EECE and fraction=1 in **case.ineec**
- **PBE:** XC\_PBE in **case.in0**. mode=EECE and fraction=1 in **case.ineec**
- **WC:** XC\_WC in **case.in0**. mode=EECE and fraction=1 in **case.ineec**
- **PBEsol:** XC\_PBESOL in **case.in0**. mode=EECE and fraction=1 in **case.ineec**
- **TPSS:** XC\_TPSS in **case.in0**. mode=EECE and fraction=1 in **case.ineec**
- **REVTPSS:** XC\_REVTPSS in **case.in0**. mode=EECE and fraction=1 in **case.ineec**
- **MBJ:** XC\_MBJ in **case.in0**. mode=EECE and fraction= $\alpha$  in **case.ineec**. 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.ineec** is necessary to start a calculation. You may copy a template from **SRC.templates** or use **init\_orb\_lapw -eece**, 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.ineec -----
-9.0 2      emin, natorb
1 1 2      1st atom index, nlorb, lorb
2 1 2      2nd atom index, nlorb, lorb
HYBR      HYBR / EECE mode
0.25      fraction of exact exchange
----- bottom of file -----
```

Interpretive comments on this file are as follows:

**line 1:** free format  
emin, natom

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

**line 2:** free format

iatom(i), nlorb(i), (lorb(li,i), li=1,nlorb(i))

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

*2<sup>nd</sup>* **line repeated natorb-times**

**line 3:** free format

mode

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

**line 4:** free format

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

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  $V_{xc}(R_{mt})$  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.

It is also possible to combine orbital potentials with the effect of an external magnetic field. In this case, you need an additional input file, **case.inorb.Bext**, which is identical to the usual **case.inorb** for external fields (see section 7.5.3). You can run this using:

► `runsp_lapw -eeceext [-so]`

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

The onsite exact-exchange/hybrid functionals from 4.5.8 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:

$$E_{xc}^{\text{hybrid}} = E_{xc}^{\text{SL}} + \alpha (E_x^{\text{HF}} - E_x^{\text{SL}})$$

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 2nd 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 lead to slower calculations. With **-mode2**, the first part of the main subroutine is parallelized over the matrix elements of the 2nd variational Hamiltonian (roughly **nband\*nband/2** where **nband** is specified in **case.inhf**, see 7.7.1), 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.vectorrhf**, **case.energyhf** and **case.weighthf** are also saved when **save\_lapw** is executed. If you restart a calculation without **case.vectorrhf**, 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.

### The available functionals

Among the semilocal functionals  $E_{xc}^{\text{SL}}$  available in **WIEN2k**, only a few of them can be used in  $E_{xc}^{\text{hybrid}}$  (both in unscreened and screened modes). The functionals are the following (**case.in0\_grr** is for the exchange part  $E_x^{\text{SL}}$ ):

- ▶ **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 7.7).
- ▶ **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 Yuakawa 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 4.5.8 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 `run_lapw -hf`

The flow of programs during a scf iteration when executing `run_lapw -hf` is the following (non-spin-polarized):

- ▶ `x lapw0 -grr` (semilocal exchange)
- ▶ `x lapw0` (semilocal exchange-correlation)
- ▶ `x lapw1` (semilocal orbitals)
- ▶ `x lapw2` (semilocal bands)
- ▶ `mv case.vectorhf case.vectorhf_old`
- ▶ `x hf (-so)` (hybrid orbitals)
- ▶ `(x lapwso` (hybrid-SO orbitals))
- ▶ `x lapw2 -hf (-so)` (hybrid electron density and bands)
- ▶ `x lcore`
- ▶ `x mixer`

#### Self-consistent calculation

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

- ▶ Do a calculation with the underlying semilocal functional  $E_{xc}^{SL}$  (recommended but not mandatory).
- ▶ "save" the semilocal calculation.

The next steps can be done conveniently using `init_hf_lapw [-up]`:

- ▶ create `case.inhf` (`cp $WIENROOT/SRC_templates/template.inhf case.inhf`).
  - Concerning `nband`:
 

While the default values for most parameters are more or less reasonable, you must set `nband` manually. It determines the size of the 2nd variational Hamiltonian, therefore technically it needs to be set to at least the number of occupied bands plus one in order to have the calculation running. The convergence of the results with `nband` should be checked, but be aware that computing time scales as `nband`<sup>2</sup>. The value of `nband` that leads to a reasonably converged result will strongly depend on the studied system and

property. For the geometry, a minimal value (number of occupied bands plus one) may already be enough. For the electronic structure (e.g., band gap), at least a few bands per atom should be added. For properties directly related to the electron density (e.g., electric field gradient or X-ray structure factors), much higher values of **nband** (up to 2-3 times the number of occupied bands) should be used and carefully tested. Note that while EMAX in **case.in1** determines the size of the 2nd variational Hamiltonian in **lapwso** (for calculations with SO), it is not the case here for hybrid (EMAX just needs to be set to a value that is large enough to provide enough orbitals to the hf-module).

- ▶ create **case.in0\_grr** (**cp case.in0 case.in0\_grr**), this file contains:
  - a screened exchange functional (e.g., EX.SPBE VX.SPBE) for the exchange functional (see above)
  - "R2V" option (instead of "NR2V") such that the exchange potential is written to **case.r2v\_grr**
  - "KXC" (instead of "TOT") such that  $\alpha E_x^{SL}$  is printed in **case.scf0\_grr** (and **case.scf**) under the label :AEXSL
- ▶ In **case.inc** the print switch has to be "1" for all atoms such that the core orbitals are printed in **case.corewf** (you don't have to set this manually, the script **run\_lapw** will do it automatically when **-hf** is specified).
- ▶ if **nband** is large, you may have to edit **case.in1** and set EMAX to a higher value, e.g., 5 Ry (or nband when using ELPA in MPI-parallel calculations).
- ▶ Execute **run\_kgenhf\_lapw**. This generates **case.klist\_fbz**, **case.klist\_ibz**, **case.kgen\_ibz** and **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 **case.klist\_ibz** (irreducible Brillouin zone).

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

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

```
run(sp)_lapw -hf ...
```

### Spin-orbit coupling

It is possible to include spin-orbit (SO) coupling in a calculation with hybrid functionals. The orbitals generated by the hf-module will be used as basis functions in the so-module. Therefore, such a calculation consists of two successive 2nd variational procedures (**lapw1**→**hf**→**lapwso**). As for semilocal functionals, the script **init\_so\_lapw** (see section 5.2.18) has to be executed in order to initialize the SO calculation. Then, start the calculation with the options **-hf** and **-so**:

```
run(sp)_lapw -hf -so ...
```

A few important points should be noted:

- ▶ The size of the Hamiltonian in the so-module is determined by the the number of bands **nband** specified in **case.inhf** (see 7.7.2), while it was EMAX in **case.in1** for semilocal calculations.
- ▶ For a spin-polarized calculation, it is necessary to regenerate the k-mesh with "**run\_kgenhf\_lapw -so**".
- ▶ The relativistic local orbitals  $p_{1/2}$  (specified in **case.inso**) can not be used.
- ▶ It does not matter in which order the scripts **init\_hf\_lapw** and **init\_so\_lapw** are executed.
- ▶ The option **-so** can be used simultaneously with the option **-diaghf**, **-redklist**, **-nonself** or **-newklist**.

### 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 option `-so`, `-nonself` or `-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 `-so`, `-nonself` or `-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 `-so`, `-diaghf` or `-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).
- ▶ Execute `run_kgenhf_lapw` with the `-newklist` switch to move previous `*klist*` files to `*_old` and create files for the new k-mesh.
 

```
run_kgenhf_lapw -newklist [-redklist ...]
```
- ▶ Run the HF calculation with `-newklist`:
 

```
run(sp)_lapw -hf -newklist (-i 1) ...
```

Note that starting from the 2nd iteration `-newklist` is automatically switched off.
- ▶ If subsequently other calculations are done with the new k-mesh, then `-newklist` must not be used, since `case.vectorhf` should now correspond to the new k-mesh.

This option can be used simultaneously with `-so` or `-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: `-so`, `-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 `-qt1` to calculate the partial charges for band character plotting.
- ▶ Create `case.insp`.
- ▶ Execute `x spaghetti` with the switch `-hf (-so -up/dn)`.

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 `-qt1` is used, then the partial charges will be calculated (`x lapw2 -band -qt1`).

### 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 -qt1 -hf (-up/dn) (-so)`) and `tetra` for the DOS (`x tetra -hf (-up/dn) (-so)`).



#### 4.5.10 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 (see [Tran et al., 2015b, Tran et al., 2016] for reports of the implementations). 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. 4.5.11) 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 Slater or KLI, then choose option “R2V” (instead of “NR2V”) in `case.in0`. For SmBJ you need to follow the procedure explained in Sec. 4.5.11 to create the starting `case.r2v` and `case.tau` 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. 5.1.6) instead:

```
run_vnonloc_lapw (-sp) ...
```

About the input file `case.inhf` (see 7.7.2) 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. 5.1.6 for the available options and flags with `run_vnonloc_lapw`.

#### 4.5.11 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 only 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.tausum` files):

- ▶ Run a regular initialization and SCF calculation using LDA or PBE (it does not matter at all which functional you choose).
- ▶ `init_mbj_lapw`. This performs automatically the following steps:
  - Create `case.inm.tau` (`cp $WIENROOT/SRC.templates/template.inm.tau case.inm.tau`).
  - Edit `case.in0` and set "R2V" option (instead of "NR2V") such that the XC potential is written in `case.r2v`.
- ▶ Run one more iteration (use `run_lapw -NI -i 1`) to generate the required `case.r2v` and `case.tausum` files.
- ▶ "save" the LDA (or PBE) calculation.
- ▶ Run `init_mbj_lapw` again and choose mBJ (option 1). The second call (once `case.inm.tau` 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`.
- ▶ Optionally, edit `case.inm` and choose the PRATT mixing scheme, although with recent `mixer` versions this should be hardly necessary.
- ▶ 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 the average of  $\nabla\rho/\rho$  over the unit cell. Such average does not make sense for interfaces and systems with vacuum like surfaces or molecules. For a system with vacuum,

a physically meaningful way to avoid this problem is first to run a calculation for a similar bulk structure (e.g., bulk graphite for graphene monolayer), then cp `case_bulk.grr` to `case.grr` and remove `case.in0.grr` (to avoid that `case.grr` is updated). This runs mBJ with a fixed value of  $\nabla\rho/\rho$  (and therefore  $c$ ). Similarly, for interfaces run the calculations for the two bulk structures constituting the interface, and then calculate the average of the two values in `case_bulk.grr` that you write in `case.grr` for the calculation on the interface (with `case.in0.grr` removed to avoid that `case.grr` is updated). However, a better alternative is to use the local mBJ potential [Rauch et al., 2020], see Sec. 4.5.12.

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].

#### 4.5.12 Local modified Becke-Johnson potential (lmBJ) for interfaces and systems with vacuum

As mentioned in Sec. 4.5.11, the mBJ potential can not be applied to interfaces and systems with vacuum without fixing  $\nabla\rho/\rho$  to the value obtained from bulk calculations. A better alternative is to use the local mBJ potential (lmBJ) [Rauch et al., 2020], where the average of  $\nabla\rho/\rho$  is calculated locally instead of in the whole unit cell. This results in a position-dependent average of  $\nabla\rho/\rho$ , and therefore also a position-dependent  $c(\mathbf{r})$ .

The steps to perform a lmBJ calculation are basically the same as for a mBJ calculation, but with the following differences:

- ▶ Choose lmBJ (option 2) instead of mBJ when `init_mbj_lapw` is executed for the second time. This will automatically
  - Select XC.LMBJ in `case.in0`.
  - Select VX.LGRR VC.NONE in `case.in0.grr`.
  - Add the value of the threshold Wigner-Seitz radius  $r_s^{th}$  (the default value is 5 bohr) at the 5th line in `case.in0abp`.
  - Create the file `case.innlvdw`, whose 7th line contains the smearing parameter  $\sigma$  (the default value is 3.78 bohr) which determines the degree of localization for the average of  $\nabla\rho/\rho$ .

NOTE that Rauch et al. [Rauch et al., 2020] used  $r_s^{th} = 7$  bohr and A,B parameters of choice "1" in `init_mbj_lapw`.

- ▶ Run the lmBJ calculation with `-lmbj`:
 

```
run (sp) _lapw -lmbj ...
```

 such that the `nlvdw` module is executed in order to calculate the position-dependent average  $\nabla\rho/\rho$ , that is stored in `case.r2v.nonloc`.

#### Correlation estimator for DFT+U

The local mBJ method can also be used to calculate a "correlation estimator" as defined by Kalantari et al. [Kalantari et al., 2021] for a transition metal (TM) atom. This helps to decide whether DFT+U should be used for a particular atom in more complicated structures like surfaces or interfaces with itinerant and correlated TM atoms. For this you would run a lmBJ initialization as described above, but then edit `case.innlvdw` and put a smaller smearing parameter (eg. 1.78, see [Kalantari et al., 2021]) in the last line of this file. Then run

- ▶ x lapw0 -grr (creates **case.r2v.grr**)
- ▶ x nlvdw -lmbj (creates **case.r2v.nonloc**)
- ▶ cp **case.r2v.nonloc case.r2v**
- ▶ x lapw5 -exchange (creates **case.in5**, unless already present)
- ▶ edit **case.in5** and change the path of the plot:
  - put the coordinates of the position of your TM atom in the first line
  - change the number of points for plotting to "2 1 "
  - change "ADD" to "NONE"
- ▶ x lapw5 -exchange -up (creates **case.rho.onedim**)

The first number (at distance 0.0) should be the desired  $\nabla\rho/\rho$  value on the chosen atom.

### 4.5.13 GLLB-SC method

For many types of solids, the GLLB-SC method [Gritsenko et al., 1995, Kuisma et al., 2010] provides band gaps that are of similar accuracy as with hybrid functionals and *GW*, but at a cost similar to semilocal methods (see, e.g., [Tran et al., 2018] for a compilation of results). With GLLB-SC, the band gap is calculated by adding an exchange discontinuity  $\Delta_x$  to the Kohn-Sham band gap (i.e., the conduction band minimum minus the valence band maximum).

The way to use the GLLB-SC method is the following:

- ▶ In **case.in0**, specify the keywords VX\_GLLBSC and VC\_PBESOL for the exchange and correlation potentials, respectively (any another choice for the correlation potential is also possible). Since GLLB-SC is a potential which has no associated expression for the energy, choose whatever you want for EX\_SWITCH and EC\_SWITCH.
- ▶ Create **case.inm.vresp** (cp \$WIENROOT/SRC.templates/template.inm.vresp **case.inm.vresp**).
- ▶ Run the calculation with **-gllb**:
 

```
run(sp)_lapw -gllb ...
```
- ▶ Save the calculation with **save\_lapw**.
- ▶ In order to calculate  $\Delta_x$ , execute the following script:
 

```
run_deltagllb_lapw (-sp) ...
```

This script will run one iteration and then **x lapw2 -all X Y** (to calculate the density of the lowest occupied orbital) and **x lapw0**. More details about **run\_deltagllb\_lapw** can be found in Sec. 5.1.7).
- ▶ Extract the value of  $\Delta_x$  in **case.scf0** (search for the keyword :DELTA\_XC) and add it to :GAP from the previously saved calculation.

### 4.5.14 DFT-1/2 method

The DFT-1/2 method [Ferreira et al., 2008] is a fast method for band gap calculations. A standard (semi)local calculation is done with an additional correction potential  $v_S$  added to the exchange-correlation potential  $v_{xc}$ . The correction potential is the difference of the KS potential of the free atom and the half-ionized atom, multiplied by a radial atom-centered step function, and summed over all (corrected) atoms:

$$v_S = \sum_{\alpha}^{\text{atoms}} \Theta(r_c^{\alpha}) A * (v_{KS}^{\alpha}(f_{\alpha} = 0) - C * v_{KS}^{\alpha}(f_{\alpha} = -1/2)).$$

Here,  $f_{\alpha}$  is the occupation number of the ionized atomic orbital with respect to the neutral atom; normally this is the orbital which contributes the most to the valence-band maximum. The cutoff

radii  $r_c$  must be variationally determined by maximizing the band gap.  $A$  and  $C$  are the ‘amplification factor’ and the ‘correction factor’ respectively. These are 1.0 in most DFT-1/2 calculations but can be used to tune the correction. Despite the name of the correction, in some materials removing 1/4 electron is more suitable. See references [Xue et al., 2018, Doumont et al., 2019] for details.

In general, DFT-1/2 should only be used for band gap calculations. It is also not suitable for all classes of materials, see [Doumont et al., 2019].

In practice, a DFT-1/2 calculation in **WIEN2k** consists of the following steps:

- ▶ Run **x lstart** to generate the KS potential for the neutral atoms (**case.potup/dn**).
- ▶ Produce **case.inst\_half** (copy **case.inst** and edit by removing 0.5 electrons from the orbitals and atoms to be corrected). Note that the latter correction is spin-dependent, so take care of the magnetic order of your system!
- ▶ Run **x lstart -half** to generate the KS potential for the 1/2-ionized atoms (**case.potup/dn\_half**).
- ▶ Produce **case.inpd**. This file defines the cutoff function and its parameters, the atomic cutoff radii  $r_c^\alpha$ , and the ‘amplification factor’  $A$  and ‘correction factor’  $C$  respectively. An example is given below.
- ▶ Run **x dstart -half [-up/dn] ...** to generate the correction potential  $v_S$  (**case.r2v\_half/dn**)
- ▶ Run the SCF calculation: **run(sp) lapw -half ...**

Since for DFT-1/2 one should optimize the atomic cutoff radii  $r_c^\alpha$ , the workflow given above needs to be repeated several times as the band gap should be maximized with respect to  $r_c^\alpha$ . A csh-script in **\$WIENROOT/SRC\_templates/iterate\_cutoff.job** can be used as template and after adaptation to your specific case simplifies the finding of the optimal  $r_c^\alpha$  for maximizing the band gap.

The necessary input file **case.inpd** looks like:

```
----- top of file: case.inpd -----
poly_out 8      # cutoff-function [parameters]
0.00 0.00      # CUTOFF RADII [AT_1_in..AT_N_in AT_1_out..AT_N_out ]
1.00 1.00      # amplification factor, correction factor
----- bottom of file -----
```

Interpretive comments follow:

**line 1:** free format

cutoff\_function [parameters]: three variants are possible

1. **poly\_out p** from [Ferreira et al., 2008]

$$\Theta_p(r, r_{\text{out}}) = \begin{cases} \left(1 - \left(\frac{r}{r_{\text{out}}}\right)^p\right)^3 & r \leq r_{\text{out}} \\ 0 & r > r_{\text{out}} \end{cases}$$

default:  $p = 8$

2. **poly.in.poly\_out p** from [Xue et al., 2018]

$$\Theta_p(r, r_{\text{in}}, r_{\text{out}}) = \begin{cases} \left(1 - \left[\frac{2(r-r_{\text{in}})}{r_{\text{out}}-r_{\text{in}}} - 1\right]^p\right)^3 & r_{\text{in}} < r < r_{\text{out}} \\ 0 & \text{else} \end{cases}$$

default:  $p = 20$

**line 2:** free format

cutoff-  
radii                      Cutoff radii for all atoms (also non-corrected ones)  
inner radii first for shell-like cutoff functions

$$\Theta(r, r_{\text{out}}) \quad r_{\text{out}}^1 \cdots r_{\text{out}}^N$$

$$\Theta(r, r_{\text{in}}, r_{\text{out}}) \quad r_{\text{in}}^1 \cdots r_{\text{in}}^N r_{\text{out}}^1 \cdots r_{\text{out}}^N$$

**line 3:** free format

amplification factor	Multiplies the total correction potential by real number (as defined above)
correction factor	Multiplies 1/2-ionized part of potential by real number (as defined above)

#### 4.5.15 DFT-D3 and DFT-D4 for dispersion energy

**dftd3** and **dftd4** calculate the dispersion energy and forces using the atom pairwise methods DFT-D3 [Grimme et al., 2010, Grimme et al., 2011] and DFT-D4 [Caldeweyher et al., 2017, Caldeweyher et al., 2019, Caldeweyher et al., 2020], respectively. Since these methods depend only on the positions of atoms (no dependence on the electron density) they are very fast and add very little computer time. The **dftd3** and **dftd4** packages are not included by default in **WIEN2k**, but can be downloaded from the website of the group of S. Grimme (<https://www.chemiebn.uni-bonn.de/pctc/mulliken-center/software>). When compilation is done (for **dftd4i** there is also a precompiled binary available), the executables **dftd3** and **dftd4** have to be copied in the **\$WIENROOT** directory.

**run(sp) \_lapw** has to be executed with the switch **-dftd3** or **-dftd4**:

► **run(sp)\_lapw -dftd3/dftd4 ...**

The user can either create the input file **case.indftd3/case.indftd4** (described in Secs. 7.2.2 and 7.3.2) by hand or let **run(sp)\_lapw** copy the default one from **\$WIENROOT/SRC.templates/**.

The **dftd3** and **dftd4** packages require the file **case.poscar** (or **case.xyz** if periodic boundary conditions are switched off) created by the utility program **struct2poscar**, which is executed automatically by **run(sp)\_lapw**.

The DFT-D3 and DFT-D4 methods contain parameters which are specific to the exchange-correlation functional to which the dispersion energy is added.

More detailed information on DFT-D3 and DFT-D4 and on the available options are given in Secs. 7.2 and 7.3 and in the manuals of **dftd3** and **dftd4**.

#### 4.5.16 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. 7.4.2) 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  $G_{\max}$  (25 bohr<sup>-1</sup>) and density cutoff  $\rho_c$  (0.3 bohr<sup>-3</sup>) should be rather well converged in most cases. However, for very weakly bound van der Waals systems, it may be safer to check the convergence by repeating the calculation with a larger  $\rho_c$  (e.g., 0.6 bohr<sup>-3</sup>). Be aware that with such a larger value for  $\rho_c$  it may be necessary to increase  $G_{\max}$  to a larger value (e.g., 40 bohr<sup>-1</sup>). More details concerning the convergence with respect to  $\rho_c$  and  $G_{\max}$  can be found in [Tran et al., 2017].
- ▶ Since the total energy changes for vdW-systems are very small, extremely high precision is necessary. Most likely, one has to increase RKmax and GMAX (`case.in2`, as well as the precision in lapw0 (`case.in0`: IFFT-parameters to -1 -1 -1 and IFFTfactor to 4 (or even 6)).
- ▶ 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 and/or OpenMP, which consists mainly of MPI/OMP FFTW (define the keyword NLVDW: in .machines, see Sec. 5.5).
- ▶ 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 `$WIENROOT/SRC_nlvdw/vdW.kernel.table` (for kernel type 1, generated with `Nr_points=2000, Nqs=30` and `q_cut=10`, which is more or less similar as in [Klimeš et al., 2011] and [Tran et al., 2017])  
`$WIENROOT/SRC_nlvdw/rVV10.kernel.table` (for kernel types 2 and 3, 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_nlvdw/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 $WIENROOT/SRC_nlvdw/
- cp vdW.kernel.table_Nr_points1024_Nqs20_q_cut5 vdW.kernel.table
```
- ▶ Kernel tables with other parameters can be generated by using the stand-alone programs `generate_vdW.kernel.table.f` and `generate_rVV10.kernel.table.f` that are in `$WIENROOT/SRC_nlvdw/`. 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. 7.4.2), 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  $Z_{ab} = -0.8491$
- ▶ 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 VX\_OPTB86B 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] (probably one of the most general vdW-functionals, which treats both, strongly and weakly bound systems and geometry and binding energies reasonably well) (see [Tran et al., 2019]):
  - `case.in0` : XC\_GGA\_X\_B86\_R EC\_LDA VC\_LDA
  - `case.innlvdw` : kernel type 1 with  $Z_{ab} = -1.887$
- ▶ SCAN+rVV10 [Peng et al., 2016]:
  - `case.in0` : XC\_SCAN
  - `case.innlvdw` : kernel type 2 with  $b = 15.7$  and  $C = 0.0093$
- ▶ PBE+rVV10L [Peng and Perdew, 2017]:
  - `case.in0` : XC\_PBE
  - `case.innlvdw` : kernel type 2 with  $b = 10$  and  $C = 0.0093$
- ▶ PBEsol+rVV10s [Terentjev et al., 2018]:
  - `case.in0` : XC\_PBESOL
  - `case.innlvdw` : kernel type 3 with  $b = 10$ ,  $C_0 = 0.0093$ ,  $C_1 = 0.5$ ,  $C_2 = 300$

#### 4.5.17 Self-consistent gKS MGGA

It is possible to perform self-consistent MGGA calculations using the generalised Kohn-Sham (gKS) scheme. For implementation details, see [Doumont et al., 2022]. It is required to compile **WIEN2k** with LibXC (see section 11.1.1) to use self-consistent MGGAs.

Note that the `XC_SCAN` or `XC_TPSS` keywords *do not* use the self-consistent gKS MGGAs. They evaluate only the ground state energy using the MGGA, using the KS orbitals corresponding to the PBE functional. Only using LibXC keywords `XC_MGGA_X/C_????` will lead to a self-consistent gKS MGGA calculation.



For the MGGAs listed in table 4.23, the setup is automatic via the `init_mgga` script. It will perform the following steps:

- ▶ If it is not yet present, create a `case.inm.tau` file.
- ▶ Prompt the user to run one or more iterations (`run(sp) -i ...`) to generate the kinetic energy density ‘tau’ required for MGGAs, and save the case.
- ▶ Set the LibXC keywords for the chosen functional in `case.in0`. In general we recommend the regularized version of SCAN, i.e. **R2SCAN**.
- ▶ Create a `case.in0.loc.vsp` input file using the appropriate (best known) auxiliary KS potential (used for core and basis functions).
- ▶ If necessary, raise the **GMAX** parameter to the suggested value.

Functional family	Auxiliary potential	Auxiliary potential keywords
SCAN family (r2SCAN, r4SCAN, SCAN)	RPBE	VX.RPBE VX.RPBE
KTBM family (KTBM.0-24)	RPBE	VX.RPBE VX.RPBE
TPSS family (revTPSS, TPSS)	RPBE	VX.RPBE VX.RPBE
other family (B97MV, regTM)	RPBE	VX.RPBE VX.RPBE
MGGAC family (rMGGAC, MGGAC)	RPBE	VX.RPBE VX.RPBE
TASK family (mTASK, TASK)	HCTH/407	VX.HCTH407 VC.HCTH407
HLE17	mRPBE	VX.MRPBE VC.MRPBE

Table 4.23: MGGAs functionals with support for automatic setup. For an exhaustive and up-to-date list of all keywords, see the LibXC website <https://libxc.gitlab.io/functionals/>.

An alternative initialization is possible using `init_mgga -taustart -dstart`. In this case a superposition of atomic kinetic energy densities is used as a starting point. This is the method of choice if you want to do a gKS calculation without a prior GGA (PBE) run. (Right after `init_lapw` do `init_mgga_lapw -taustart`)

For computationally intense cases, the increase of **GMAX** can be overridden using the flag `-skipgmax`. The suggested value that is set by the script is a quite ‘safe’ value, and for most cases not necessary. At the same time, if very high convergence is required one might still check if the suggested value is sufficient.

If **QTL-B** warnings, or ‘Semicore band ranges too large’ errors may occur, `init_mgga -hdlo` may be used to set extra HDLO basis functions. This solves the problem in many cases.

For functionals that are not listed in the `init_mgga` script, it will guide you through a manual setup, which consists of the same steps outlined above. The difference is that the user is prompted to edit the relevant input files manually (like the general `init` script). For readers interested in the manual setup, we strongly recommend to read section 2 of the implementation paper [Doumont et al., 2022]. A few hints for the manual setup:

1. A current list of MGGAs included in LibXC can be found online on <https://libxc.gitlab.io/>.
2. The self-consistent gKS MGGAs implementation only works with ‘tau’-MGGAs, not with laplacian-MGGAs. For a laplacian-MGGAs, an error will be given.
3. Some MGGAs *potentials* do not have an associated energy functional (e.g. (modified) Becke-Johnson). For these, the gKS implementation is not applicable.
4. If the functional you want to use is not included in the script, we are not sure (yet) what is the best choice for the auxiliary potential in `case.in0.loc.vsp` is. RPBE (**EX.RPBE EC.PBE VX.RPBE VC.PBE**) or PBE **XC.PBE** are probably a good choice.
5. The best auxiliary potential is the one leading to the lowest variational total energy :**ENE**.
6. Use of HDLOs (see section 7.6) is recommended in case you are not sure what is the best auxiliary potential.
7. Take care that there are no large **QTL-B** warnings in the scf files.

In some cases (specifically cases with vacuum regions in the cell, like surfaces or monolayers), and with some functionals that are more difficult numerically (like non-regularized SCAN or TASK) scf-convergence can be tricky. The following steps may help:

- ▶ Start from a well-converged KS calculation, if possible one with similar characteristics to the MGGA (e.g. PBE for SCAN). Starting from a well-converged gKS calculation with a numerically less-demanding MGGA (e.g. a regularized SCAN) may also work.
- ▶ Make sure energy parameters are correctly set. In some cases, shallow LOs (like O-2p) and spheres with small  $R_{mt}$  can be problematic. Replacing them with HDLO can help. In rare cases, RKmax needs to be restricted (eg. in Fe less than 9), otherwise ghostbands can occur.
- ▶ Make sure sphere sizes are properly chosen.
- ▶ If all else fails, use 'PRATT' mixing for both the density and the kinetic energy density, for at least a couple of iterations, with the same mixing parameter (see section 7.13).

Forces are not available with gKS MGGA calculations.

---

# 5 Shell scripts for running programs

---

## Contents

---

5.1	Job control . . . . .	71
5.2	Utility scripts . . . . .	80
5.3	Structure optimization . . . . .	89
5.4	Phonon calculations . . . . .	96
5.5	Parallel Execution . . . . .	98
5.6	NMR calculations: Chemical shift and Knight shift . . . . .	106
5.7	Wannier functions (wien2wannier) . . . . .	112
5.8	Spontaneous Polarization, Piezoelectricity and Born Charges, Weyl points (BerryPI) . . . . .	114
5.9	Getting on-line help . . . . .	117
5.10	Interface scripts . . . . .	118

---

## 5.1 Job control (c-shell scripts)

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

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

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

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

### 5.1.1 Main execution script (x\_lapw)

The main **WIEN2k**script, **x\_lapw** or in short just **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 tailoring special behaviour. All options are listed with the help switch

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

With some of the options the corresponding input files may be changed temporarily, but are set back to the original state upon completion. If a necessary input file does not exist for auxilliary programs, it will be created with some defaults and you should edit and adapt this file for your needs.

USAGE: x PROGRAMNAME [flags]

PURPOSE: runs WIEN executables: afminput, afmsim, aim, animssf, arrows, broadening, cif2struct, clmaddsub, clmcopy, clminter, convham, conv2prim, dftd3, dftd4, 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, mstar, nlvdw, nn, optimize, orb, pairhess, pes, plane, read\_vorb\_files, rendos, renormalize\_dmat, rhomb\_in5, sgroup, shifteig, spaghetti, struct2cif, struct2poscar, struct\_afm\_check, sumpara, supercell, symmetry, symmetso, telnes3, tetra, txspec, wannier90, w2w, w2waddsp, wplot, xspec, xyz2struct, 3ddens, create\_rho

FLAGS:

```
-f FILEHEAD -> FILEHEAD for name 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-crossterm
-sc -> runs semicore calculation
-p -> run lapw0/l/2/hf/so/dm/optic/dstart in parallel (needs .machines or
.processes file)
-scratch dir/ -> defines (and makes) $SCRATCH variable
-half -> run lstart/dstart/lapw0 for DFT-half
-hdlo -> run lstart and create p,d,f HDLOs in $file.in1_st
-grr -> lapw0 for mBJ, lmBJ or hf (using $file.in0_grr)
-eece -> for hybrid-functionals (lapw0, lapw2, mixer, orb, sumpara)
-band -> for lapw1/2/hf/joinvec bandstructures: uses *klist_band
-orb -> runs lapw1 with LDA+U/OP or B-ext correction, mixer with dmat
-it -> runs lapw1 with iterative diagonalization
-noHinv -> runs lapw1 with iterative diag. without Hinv
-noHinv0 -> runs lapw1 with iterative diag. writing new Hinv
-nohns -> runs lapw1 without HNS
-nmat_only -> runs lapw1 and yields only the matrixsize
-nmr -> runs lapw1 in NMR mode
-inlorig -> runs lapw2 but does not modify case.in1
-emin X -> runs lapw2 with EMIN=X (in $file.in2)
-all X Y -> runs lapw2 with ALL and E-window X-Y (in $file.in2)
-qt1 -> runs lapw2 and calculates QTL
-alm -> runs lapw2 and calculates ALM, BLM
-almD -> runs lapw2 and calculates ALM, BLM in lapw2 for DMFT (Aichhorn)
-qdmft -> runs lapw2 and calculates charges including DMFT (Aichhorn)
-help_files -> runs lapw2 and creates case.helpXX files
-vresp -> runs lapw2/mixer and creates case.vrespval/sum (meta-GGA)
-tau -> for lstart, dstart, lapw2, lcore and mixer, creates case.tauval/sum (meta-GGA)
-fermi -> runs lapw2 with FERMI switch
-efg -> runs lapw2 with EFG switch
-so -> runs lapw2/dm/optic/spaghetti with def-file for spin-orbit calc.
-hf -> runs lapw2 with Hartree-Fock/hybrid vectors
-diaghf -> calculates only the diagonal elements of HF Hamiltonian
-nonsel -> calculates hf with Ex only (no eigenvalues/vectors)
-model2/3 -> modes 2 and 3 calculate hf with a better MPI scaling for memory
-nonsel -> calculates hf with Ex only (no eigenvalues/vectors)
-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
-kli -> calculation of the KLI potential by the HF module
-gllb -> calculation of the GLLB-SC potential
-gw -> write in case.gw the diagonal matrix elements of the HF/hybrid Hamiltonian for GW
-fbz -> runs kgen and generates a full mesh in the BZ
-fft -> runs dstart only up to case.in0_std creation
-super -> runs dstart and creates new_super.clmsum (and not $file.clmsum)
-lcore -> runs dstart with $file.rsplcore (produces $file.clmsc)
-val -> runs lapw5/3ddens with normalization factor for files case.clmval, vttotal, vcoul, r2v
-tot -> runs lapw5/3ddens with normalization factor for files case.clmsum
-pot -> runs lapw5/3ddens with vttotal-files as input
-coulomb -> runs lapw5/3ddens with vcoul-file as input
-exchange -> runs lapw5/3ddens with r2v-files as input
-exchange2 -> runs lapw5/3ddens with r2v2-files as input
-half2v -> runs lapw5/3ddens with r2v_half-files as input
-diff -> replaces "RHO" by "DIFF" in case.in5 for difference density
-sub -> replaces "add" by "sub" in case.in3d/in5 for spin-density plots
-add -> replaces "sub" by "add" in case.in3d/in5 for density plots
-none -> replaces "add" by "non" in case.in3d/in5 for up-density plots
-sel -> use reduced vector file in lapw7
```

```

- settol 0.000x -> run sgroup, mstar or xyz2struct with different tolerance
- sigma->       run lstart with case.inst_sigma (autogenerated) for diff.dens.
- dftd3->       run mixer with dftd3 forces
- dftd4->       run mixer with dftd4 forces
- nlvdw ->      run lapw0 with -nlvdw flag (adding the vdW-potential)
- lmbj ->       run nlvdw with -lmbj flag (creates local g for lmbj potential)
- rxes->       run tetra using case.rxes weight file for RXES-spectroscopy.
- rxesw E1 E2-> run tetra and create case.rxes file for RXES for energies E1-E2
- enefile ->    spaghetti+tetra with case.energy instead case.qtl (only tot-DOS)
                joinvec: only join case.energy_* files
- delta->      run arrows program with difference between two structures
- copy ->     runs pairhess and copies .minpair to .minrestart and .minhess
- telnes ->    run qtl after generating case.inq based on case.innes
- xspec ->    run broadening with case.xspec instead of case.elnes
- pes ->     run broadening with case.pes1 instead of case.elnes
- txt ->     runs cif2struct using case.txt (see UG)
- pp ->     run wannier90 in "preprocessing mode"
- wf N ->    run wplot for Wannier function N
- efermi EF -> run findbands (unit:Ryd) / shifteig (unit:eV) with Fermi energy EF
- emax Y ->   for findbands
- job command -> for optimize, run-command in optimize.job
- save name -> for optimize, appended savefilename in optimize.job (instead of default)
USE: x -h PROGRAMNAME for valid flags for a specific program

```

*Note: To make use of a scratch file system (usually a "local" file system for reducing the network or central fileservers 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 directory and run all calculations from there. At the beginning one must provide at least one file, namely **case.struct** (see 4.3). This can be created in **w2web** (see quick-start, Sec. 3), the **makestruct\_lapw** script (see 5.1.2), or from a cif, xyz or poscar file (see 9.7 and 9.32).

The script **init\_lapw** runs a series of programs either step by step (when using the -m switch), or - highly recommended - in batch mode (default). The necessary **case.inst** will be created automatically on the "fly", only for spin-polarized calculations with special magnetic order (e.g. antiferromagnetic) it must first be created manually by **instgen\_lapw -ask** ( see 6.4.3).

**init\_lapw** is described briefly in chapter 4.5 and the manual step-by-step mode in detail in "Getting started" for the example of TiC (see chapter 3).

As mentioned above, the **batch-mode** is highly recommended (and now the default), since it sets (semi-)automatically many computational parameters in the various input files, which can be customized using several switches. You can get help about all of them with switch -h.

```
init_lapw -h
```

```

PROGRAM: /zeus/WIEN2k/init_lapw
PURPOSE: initialisation of WIEN2k calculations
        to be called within the case-directory
        needs case.struct file
USAGE:  init_lapw [OPTIONS] [FLAGS]
FLAGS:
-h/-H  -> help
-m     -> manual step-by-step mode (not recommended anymore)
-b     -> batch (non-interactive) mode (default, all possible options are
        listed below, SGROUP is always ignored)
-sp    -> select spin-polarized calculation
-nodstart -> creates new input files, but no case.clmsum (assuming you
        already have a converged calculation)
-nokshift -> produces an unshifted k-mesh (including Gamma)
-nometal-> reduces k-mesh by factor 10
-hdlo  -> set HDLOs in lstart (case.inl)
-nohdlo -> do not set HDLOs in lstart (case.inl) (overwrites -prec 2/3 setting)

OPTIONS:
-f FILEHEAD -> FILEHEAD for name of struct & input-files
-prec X     -> set precision 0/1/2/3/0n/1n/2n/3n (default:1; "n" means
        no metal for reduced k-mesh)
-red X      -> RMT reduction by X % or with X=Si:2.0,O:1.6 (default: RMT not changed)
-vxc X      -> VXC option (default: PBE; (LDA, WC, PBESOL))
-fftfac X   -> Enhancement factor of FFT grid (default: automatic)
-fft X Y Z  -> sets FFT grid to X Y Z (grid dimensions in lapw0, default: automatic)
-autofft    -> sets FFT grid to -1 -1 -1 (grid determined by lapw0)
-ecut X     -> energy separation (or Q/sphere) between core/valence (default: -6.0 Ry)
-rkmax X    -> RKMAX (default: automatic)
-lmax X     -> LMAX (default: 10)
-lvns X     -> LVNS_max (default: automatic)
-gmax X     -> GMAX (default: automatic)
-fermit X   -> use TEMP with smearing by X Ry (default: TETRA)
-fermits X -> use TEMPS with smearing by X Ry (default: TETRA)
-numk X     -> use X k-points in full BZ (default: automatic);
        or: 0 NX NY NZ (with unshifted mesh) or: -1 delta-K
-s PROGRAM  -> in manual mode: start with PROGRAM ($next)
-e PROGRAM  -> in manual mode: exit after PROGRAM ($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 manual mode (-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).

**init\_lapw** runs by default in “batch” mode (non-interactive). We recommend the batch mode in all cases once the symmetry and equivalency of atomic positions are correct for a new case (eventually run **x nn**, **sgroup**, **symmetry** first, accept eventually the modifications of the **case.struct** file and make sure the symmetry is correct before starting the initialization in batch mode, because changes to **case.struct** by **nn** will be accepted, but by **sgroup** will be neglected).

Using **-prec X** (default prec=1) the script sets all input parameters according to the selected precision (select “n” when you know that your case is non-metallic for reduced k-mesh).

- ▶ 0/0n: for big (surface) calculations and possible long position optimizations. Should be checked later on.
- ▶ 1/1n: default, should be safe for bands or DOS
- ▶ 2/2n: highly accurate also for total energies, reduces  $RMT_{max}$  to 2.35, sets HDLOs and avoids core-leakage
- ▶ 3/3n: highest precision, for lattice parameters with 3 digits after the comma

You can supply various options and select spin-polarization or XC-potential. The automatic settings for RKmax, l-max, Lvns-max (nonspherical matrix elements for large spheres), HDLOs, GMAX, k-mesh, FFT-grids in lapw0 or the “Fermi-method” can be overwritten by the corresponding options. For 2D cases TEMP instead of TETRA will be selected automatically.

Once you have a scf solution (and have "saved" the results), you can rerun `init_lapw` with higher precision and the `-nodstart` switch, to create a more accurate input (but continue the scf with the previously converged density). However, when you have large RMTs, using `-prec 2/3` may reduce the atomic spheres. In such a case you cannot use the previous density and the `-nodstart` switch.

*Please check the terminal output for ERRORS and WARNINGS !!!*

### 5.1.4 Job control for scf-iterations (run\_lapw or runsp\_lapw)

In order to perform a complete SCF calculation or even perform both, a 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 you can use: **runsp\_lapw** or **runafm\_lapw** (the latter requires an additional (complicated) input file, but can save some cpu-time)
- ▶ for FSM (fixed-spin moment) calculations use: **runfsm\_lapw**
- ▶ for a spin-polarized setup, where you want to constrain the moment to zero (e.g. for LDA+U calculations) use: **runsp\_c\_lapw**

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

#### **run\_lapw -h**

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

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

USAGE:            run_lapw [OPTIONS] [FLAGS]

OPTIONS:
-cc LIMIT ->     charge convergence LIMIT (0.0001 e)
-ec LIMIT ->     energy convergence LIMIT (0.0001 Ry)
-fc LIMIT ->     force convergence LIMIT (1.0 mRy/a.u.)
.str LIMIT->     stress convergence LIMIT (0.1 GPa)
                  default is -ec 0.0001; multiple convergence tests possible
-e PROGRAM ->    exit after PROGRAM ()
-f FILEHEAD ->  alternative FILEHEAD for all files (instead of dir name)
-i NUMBER ->    max. NUMBER (40) of iterations
-s PROGRAM ->   start with PROGRAM ()
-r NUMBER ->    restart after NUMBER (99) iterations (touch .restart)
-fd NUMBER ->   force full diag after NUMBER iterations (touch .fulldiag)
-nohs NUMBER ->do not use HNS for NUMBER iterations
-inlnew N ->    create "new" inl file after N iter (write_inl using scf2 info)
-ql LIMIT ->    select LIMIT (0.05) as min.charge for E-L setting in new inl
-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 [speed:name])
-it ->          use iterative diagonalizations
-it1 ->         use iterative diag. with recreating H_inv (after basis change)
-it2 ->         use iterative diag. with reinitialization (after basis change)
-noHinv ->      use iterative diag. without H_inv
-vec2pratt ->   use vec2pratt instead of vec2old for iterative diag.
```

```

-grid ->      test grid topologies (1/2 iteration) and use the faster one
              (only for MPI-parallel with non-squared grids)
-so ->       run SCF including spin-orbit coupling
-renorm->    start with mixer and renormalize density
-inlorig->   if present, use case.inl_orig file; do not modify case.inl
-half ->     DFT-half calculation
-hf ->      HF/hybrid-DFT calculation
-diaghf ->  non-selfconsistent HF with diagonal HF only (only e_i)
-model1/2/3 -> modes 2 and 3 calculate hf with a better MPI scaling for memory
-nonsel ->  non-selfconsistent HF/hybrid-DFT calculation (only E_x(HF))
-newklist -> HF/hybrid-DFT calculation starting from a different k-mesh
-redklist -> HF/hybrid-DFT calculation with a reduced k-mesh for the potential
-slater->   calculation of the Slater potential by the HF module
-kli ->    calculation of the KLI potential by the HF module
-gllb ->   calculation of the GLLB-SC potential
-gw ->     write in case.gw the diagonal matrix elements of the HF/hybrid
           Hamiltonian for GW
-dftd3 ->  calculate the dispersion energy with the DFT-D2 or DFT-D3 method
-dftd4 ->  calculate the dispersion energy with the DFT-D4 method
-nlvdw ->  include corrections due to nonlocal van der Waals functional
-lmbj ->   to calculate the lmbj potential
-min ->    force optimization using MSRLa

CONTROL FILES:
.lcore      runs core density superposition producing case.clmsc
.stop      stop after SCF cycle
.minstop    in MSRLa mode(structure optimization) switches to MSRL
.minstart   starts MSRLa minimization during scf run (deletes broyd* files)
.fullldiag force full diagonalization
.noHinv     remove case.storeHinv files
case.inm_tau activates calculation of tau files for meta-GGAs
case.inm_vresp activates calculation of vresp files for meta-GGAs
case.in0_grr activates a second call of lapw0 (mBJ pot., or E_xc analysis)

ENVIRONMENT VARIABLES:
SCRATCH     directory where vectors and help files should go

```

Additional flags valid only for magnetic cases (**runsp\_lapw**) include:

```

-dm ->      calculate the density matrix (when -so is set, but -orb is not)
-eece ->    use "exact exchange+hybrid for correlated electrons" 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)
-orbext ->  use LDA+U and B-ext corrections simultaneously
-eeceext -> use EECE and B-ext corrections simultaneously

.forceorb   uses unmixed case.vorb* in next iter (-eece)
.forcedmat  uses unmixed case.dmat* in next iter (-orb)

```

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 four 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!)
- ▶ **-cc** (charge convergence, just the last iteration),
- ▶ **-str** (stress convergence, for details see Sect. 5.3.1 )
- ▶ 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 another **run\_lapw** without "**save\_lapw**" (because the previous run did not fulfill the convergence



criteria or you want to specify a more strict criterium) the "broyden-files" will be deleted unless you specify `-NI`.

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

Invoking

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

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

An additional switch `-min` will activate the optimization of the internal positions using the `MSR1a` option in `case.inm` (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 " $E_F - 0.2$ ". The switch `-inlorig` can be used to keep the present `case.in1` file unmodified (or to copy `case.in1_orig` back after `-in1new`).

The switch `-in1new N` preserves for N iteration the current `case.in1` file. After the first N iterations `write_in1_lapw` is called and a new `case.in1` file is generated, where the energy parameters are set according to the `:EPLxx` and `:EPHxx` values of the last scf iteration and the `-ql` value (see sections 4.4 and 7.6). 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 `case.in1` file is saved in `case.in1_orig` and is used as template for all further scf-cycles.

Parallelization is described in Sec. 5.5.

Iterative diagonalization, which can significantly save computer time (in particular for cases with "few electrons" (like surfaces) and "large matrices (larger than 2000)" a factor 2-5 ! is possible), is described in Sec. 7.6. It needs the `case.vector_old` file from the previous scf-iteration (and this file is created from `case.vector` when the `-it` switch is set) and an inverse of a previous Hamiltonian ( $H_0^{-1}$ ) stored in `case.storeHinv`. When you change the Hamiltonian significantly (changing RKmax or local orbitals), reinitialize the iterative diagonalization either by "`touch .fulldiag`" (performs one full diagonalization) or "`touch .noHinv`" (recreates `case.storeHinv` files) or using the `-it1|-it2` switch.

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

The presence of the file `.lcore` directs the script to superpose the radial core densities using `dstart` and generating `case.clmsc`. It is created automatically during `init_lapw` when charge-leakage warnings are ignored. This option allows to reduce the number of semi-core states, but still keeping a good charge density. `dstart` can also run in mpi- or OpenMP-parallel mode, otherwise it can be slow for big cases.

The presence of the file `case.in0_grr` activates a second call of `lapw0`, which is necessary for modified Becke-Johnson potentials (see Section 4.5.11) 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 `-hf` switch. More information can be found in Sec. 4.5.9

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

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

### 5.1.5 Job for initialization and scf-cycle with different precision (run123.lapw)

This script performs an initialization (`init.lapw`) and scf-cycle (`run.lapw`) with different precision options. It only needs a `case.struct` file as input. You can always get help on its usage by invoking this script with the `-h` flag.

```
run123.lapw -h
```

```
run123_lapw performs init_lapw and run_lapw with different precision
```

```
run123_lapw [ runopt initopt -noprec [1-2] -prec [1-3] -numk2 k-mesh2-params -h ]
```

```
It runs scf cycles with precision options from:
```

```
-noprec XX+1 (default 0) up to -prec XX (default 3)
```

```
and saves them under prec1, prec2, prec3 (prec3k)
```

```
runopt: all valid options for run_lapw (like -p, -i 100, -ec 0.000001, ..)
```

```
initopt: only supported init_lapw options:
```

```
(-red X, -hdlo, -nokshift, -fermit(s) 0.00x, -numk XX, -fft X Y Z)
```

```
-numk2 k-mesh-params reruns prec3 with different (better) k-mesh
```

In parallel mode (`-p`) it will either use a `.machines` file (if present) or use `$WIENROOT/SRC.templates/run123.machines`, which is configured for one 8-core cpu.

### 5.1.6 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. 4.5.10) it is necessary to use the script `run_vnonloc.lapw`. 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 (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 `run(sp).lapw` (`-ec`, `-cc`, etc.) can be used with `run_vnonloc.lapw`. All options and flags can be listed with the `-h` flag:

```
run_vnonloc.lapw -h
```

```
PROGRAM: run_vnonloc_lapw
```

```
PURPOSE: to run the inner/outer loops procedure to achieve scf convergence with the Slater/SMBJ/KLI potentials
```

```
USAGE: run_vnonloc_lapw [OPTIONS] [FLAGS]
```

```

OPTIONS:
-cc LIMIT -> charge convergence LIMIT (0.0001 e)
-ec LIMIT -> energy convergence LIMIT (Ry)
-fc LIMIT -> force convergence LIMIT (1.0 mRy/a.u.)
              default is -ec 0.0001; multiple convergence tests possible
-i NUMBER -> max. NUMBER (40 by default) of iterations for the inner loop
-iout NUMBER -> max. NUMBER (100 by default) of iterations for the outer loop
-inlnew N -> create "new" inl file after N iter (write_inl using scf2 info)
-mix -> use another mixing factor for the outer loop (default is 1)

FLAGS:
-h/-H -> help
-so -> run SCF including spin-orbit coupling
-sp -> for spin-polarized calculation
-NI -> does NOT remove case.broyd* (default: rm *.broyd* after 60 sec)
-p -> run k-points in parallel (needs .machine file [speed:name])
-inlorig-> if present, use case.inl_orig file; do not modify case.inl

```

### 5.1.7 Job control for calculating the exchange discontinuity of the GLLB-SC method (run\_deltagllb\_lapw)

The script `run_deltagllb_lapw` allows to calculate the exchange discontinuity  $\Delta_x$  of the GLLB-SC method (see sec. 4.5.13) in a convenient way. First, one iteration is done, and then the lowest occupied orbital is calculated with `x lapw2 -a11 X Y`, and finally `x lapw0` is executed to calculate  $\Delta_x$  (keyword :DELTA<sub>X</sub> in `case.scf0`).

For spin-polarized calculations, the flag `-sp` has to be used. Most of the options and flags of `run(sp)_lapw` (except `-ec`, `-cc`, and `-fc`) can be used with `run_deltagllb_lapw`. All options and flags can be listed with the `-h` flag:

#### `run_deltagllb_lapw -h`

```

PROGRAM:      run_deltagllb_lapw

PURPOSE:      to run one iteration in order to calculate the derivative
              discontinuity of the GLLB-SC potential

USAGE:        run_deltagllb_lapw [OPTIONS] [FLAGS]

OPTIONS:
-inlnew N -> create "new" inl file after N iter (write_inl using scf2 info)
-scratch dir -> set scratch directory (for vector files)

FLAGS:
-h/-H -> help
-sp -> for spin-polarized calculation
-p -> run k-points in parallel (needs .machine file [speed:name])
-it -> use iterative diagonalization
-it1 -> use iterative diag. with recreating H_inv (after basis change)
-it2 -> use iterative diag. with reinitialization (after basis change)
-noHinv -> use iterative diag. without H_inv
-so -> run SCF including spin-orbit coupling
-inlorig-> if present, use case.inl_orig file; do not modify case.inl

```

## 5.2 Utility scripts

### 5.2.1 Save a calculation (save\_lapw)

After self-consistency has been reached, the script

```
save_lapw head_of_save_filename
```

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

For more complicated situations, where many parameters will be changed, we have extended **save\_lapw** so that calculations can not only be saved under the **head\_of\_save\_filename** but also a directory can be specified. 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
-nodel     do not delete scf, broyd files (during running scf-cycle)
-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
-xspec     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 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

### 5.2.4 Migrate a case to/from a remote computer (migrate\_lapw)

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

Call it within the desired case-dir as:

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

with the following options:

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

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

FLAGS:

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

### 5.2.5 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 (default: directory name). 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 use 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.

### 5.2.6 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 -f case]
-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
-f:   case name (instead of directory name)
```

### 5.2.7 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).

### 5.2.8 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.

### 5.2.9 grepline\_lapw

This scripts allows you to get a certain quantity from several scf files for comparison (for instance the total energy :ENE in the saved scf-files of a volume-optimization calculation). Using

```
grepline_lapw :label 'filename*.scf' lines_for_tail [options]  or
grepline :label 'filename*.scf' lines_for_tail [options]
```

you can get a list of a quantity "**:label**" (e.g. :ENE for the total energy, :DIS, :FR, :FGLxxx, :MMT, :MMIxxx, ... are other useful possible labels) from several scf files at once. Specification of "lines\_for\_tail" .gt. 1 allows for convergence check too.

"options" can be -h (help), -s (silent and avoiding the first line), -s0 (silent) and grep options like -B1 or -A1 (usefull to grep for :MBJ and list the line afterwards).

### 5.2.10 scfmonitor\_lapw

This program was contributed by:

⇒ Hartmut Enkisch  
 Institute of Physics E1b  
 University of Dortmund  
 Dortmund, Germany

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 GNU PLOT. This plot is updated in regular intervals.

You can call **scfmonitor\_lapw** using:

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

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

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

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

### 5.2.11 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:  :ENE :DIS :FER :MMT :VOL :GAP :CONSTRAINT
atom iii dependent: :CTOiii :CUPiii :CDNiii :NTOiii :NUPiii :NDNiii
                  :DFOiii :DUPiii :DDNiii :RTOiii :EFGiii :HFFiii
                  :MMIiii :VCOUL
vector quantities: :FORiii[x/y/z] :POSiii[x/y/z] :FGLiii[x/y/z] :FGCiii(x/y/z)
                  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.

### 5.2.12 Extract critical points from a Bader analysis (`extractaim_lapw`)

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

It is used with: `extractaim_lapw case.outputaim`

### 5.2.13 Check parallel execution (`testpara_lapw`)

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

### 5.2.14 Check parallel execution of `lapw1` (`testpara1_lapw`)

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

### 5.2.15 Check parallel execution of `lapw2` (`testpara2_lapw`)

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

### 5.2.16 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. The original version was provided by Morteza Jamal ([m.jamal57@yahoo.com](mailto: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)
```

At the end, band-ranges and EF are listed and you can edit `case.int` for different E-grid or DOS-summations.

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

```

### 5.2.17 `init_orb_lapw`

`init_orb_lapw` helps you to initialize calculations using orbital potentials like DFT+U and EECE with/without an additional external magnetic field (based on an idea of William Lafargue-Dit-Hauret, [william.lafargue-dit-hauret@uliege.be](mailto:william.lafargue-dit-hauret@uliege.be)). It creates all required input files (`case.inorb`, `case.indm`, `case.ineece`) and allows to modify them according to your needs (specification of atoms, orbitals, U+J values). It is called using

```

init_orb_lapw -orb / -eece [-b -f -c] or
init_orb -orb / -eece [-b -f -c]

```

Either the option `-orb` or `-eece` is required. `-b` adds an optional external magnetic field; `-f` forces a recreation of previously existing input files; `-c` is necessary only if you have inversion symmetry, but want to include spin-orbit (-so) in the calculations (otherwise a complex case is detected automatically).

### 5.2.18 `init_so_lapw`

`init_so_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](mailto: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

```

init_so_lapw or
init_so

```

and you should *carefully* follow the instructions and explanations of the script and the explanations for `case.inso` given in section 7.8. 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  $\text{UO}_2$ ), because then one can optimize the O-positions.

### 5.2.19 `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 4.5.9.

### 5.2.20 `init_mbj_lapw`

`init_mbj_lapw` helps you to initialize the calculations for the mBJ (see [Tran and Blaha, 2009]) or lmBJ (see [Rauch et al., 2020]) potential, 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.tau` and sets "R2V" in `case.in0`. The second call of `init_mbj_lapw` creates `case.in0` and `case.in0_grr` (and `case.innlvdw` for lmBJ) with the proper input for (l)mBJ. It also lets you select parameters of the original (l)mBJ potential, or the later adaption to semiconductors or insulators, or the original BJ method.

For details of (l)mBJ calculations see Secs. 4.5.11 and 4.5.12.

### 5.2.21 `init_mgga_lapw`

`init_mgga_lapw` helps you initialize a self-consistent (gKS) MGGA calculation (see section 4.5.17). For some (popular) functionals an automatic setup is possible (e.g. SCAN or the KTBM-family (see [Kovacs et al., 2022])). Otherwise a guided manual setup is offered.

It is also possible to re-use the script after initialization to more easily change to another MGGA functional.

It may be necessary to run the script twice, with one scf iteration in between, to generate the kinetic energy density required for an MGGA (if it was not yet present). The script will notify you in this case.

Help is available with the switch `-h`.

#### `init_mgga_lapw -h`

```
PROGRAM:          init_mgga_lapw

PURPOSE:          initialisation of a self-consistent (gKS) MGGA calculation

USAGE:           init_mgga_lapw [OPTIONS] [FLAGS]

FLAGS:
-f FILEHEAD -> FILEHEAD for path of struct & input-files
-h/-H       -> help
-keepgmax   -> Do not set the suggested GMAX for the chosen MGGA potential,
               instead keep the current one.
-taustart   -> Generate the $file.tausum(up/dn) files from the atomic kinetic energy densities
-dstart     -> Regenerate the electronic density from the atomic densities (for consistency)
               (recommended if -taustart is used)
```

### 5.2.22 `vec2old_lapw`

`vec2old_lapw` moves `case.vector` files to `case.vector.old`. Usually called automatically just before `lapw1` when the iterative diagonalization (`run_lapw -it`) is specified. It also works for the k-parallel case including local `$SCRATCH` directories (add `-p` as first argument, uses hosts from `.processes` and 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 -qt1`.

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 joinvec\_lapw

`joinvec_lapw` is a script which together with the program `join_vectorfiles` joins parallel `case.vector*`, `case.energy*` and `case.energydum*` files into a single `case.vector/case.energy/case.energydum`. It will read the information about the parallel files from `.processes` (generated by the parallel `lapw1` execution).

It is executed using:

```
x joinvec [ -up/dn -so -hf -enefile]
```

The switch `-enefile` directs the program to join only the energy files but not the vector files. Using `-so` or `-hf` you can join parallel spin-orbit or hybrid-DFT (Hartree-Fock) vector files.

### 5.2.24 Reduce atomic spheres and interpolate density (reduce\_rmt\_lapw)

```
reduce_rmt_lapw [ -r XX / -a XX:Rxx,YY:Ryy, ... -sp -vxc X]
```

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` or `-a .;` i.e. same syntax as used in `set_rmt_lapw`) and interpolate the density inside the spheres to the new radial mesh. It uses internally the program `clmaddsub` to adapt the interstitial charge density. Setting the switch `-sp` will do it for `clmsum`, `clmup` and `clmdn` files. We use PBE (13) as default, but with `vxc=5, 11, 19` you can set LDA, WC or PBEsol, respectively.

### 5.2.25 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 uses internally the program `clmaddsub` which 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. However, when you also do a relaxation of internal positions, you should run `x dstart -super` after the relaxation and BEFORE new lattice parameters are introduced.

It is usually called from "`min_lapw`" or "`optimize.job`" 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.

### 5.2.26 `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 on a surface). (It uses internally `clmaddsub`, `create_add_atom_clmsum_exscript_1_lapw` and `create_add_atom_clmsum_exscript_2_lapw`).

It is useful in BIG cases, which are difficult to converge from `init_lapw`.

The following steps should be done in the directory with the converged calculation (“surface”, needs to have `clmsum/rsp` files):

- ▶ `mkdir adsorbate`
- ▶ `cd adsorbate`
- ▶ `cp ../surface.struct adsorbate.struct`
- ▶ `edit adsorbate.struct` # add an atom at the desired position as last atom
- ▶ `instgen_lapw -ask` # optional for AFM cases
- ▶ `create/copy .machines file` # optional for dstart
- ▶ `init_lapw -b [-sp ...]`
- ▶ `create_add_atom_clmsum_lapw [-p -sp]`

In cases with LDA+U you may further copy the `dmatup/dn` and `vorbup/dn` files and start with `runsp_lapw -orb` for some iterations.

## 5.3 Structure optimization

Structure optimization consists in general of at least 2 steps (the determination of the optimal crystal structure (searching for different phases like NaCl or CsCl, ... will not be discussed here); optimization of atomic positions (when not fixed by symmetry); and optimization of the lattice parameters (and angles).

If your structure has free internal position parameters, always optimize them first (see Sect. 5.3.2) and reoptimize them for every change in lattice parameters (verify that the remaining forces on the atoms are small).

Optimization of the lattice parameters can be done by the methods discussed below.

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

#### Stress

The stress tensor, defined as the derivative of the total energy  $E_{tot}$  with respect to strain  $\epsilon_{ij}$  ([Belbase et al., 2021])

$$\sigma_{ij} = \frac{1}{\Omega} \frac{dE_{tot}(\epsilon)}{d\epsilon_{ij}} \quad (5.1)$$

as well as its trace, the pressure, can in principle be used to find the equilibrium lattice parameter corresponding to zero stress (or at a certain pressure). So far, we do not provide an utility to automatically minimize the stress (optimize the lattice parameters), but you have to do it manually.

Unfortunately, the stress tensor as implemented currently has severe restrictions and can be used ONLY for non-relativistic (NREL) calculations. The accuracy is thus limited to first and second row elements (maybe early 3d elements) and in particular for the 5d series and above, large errors may occur due to the neglect of (scalar-) relativistic corrections. It is also necessary to use a XC-potential from the LIBXC-library, so `lapw0` must be compiled using the LIBXC library. Finally, this option is fairly expensive (see below) and needs a rather large RKMAX (`init.lapw -prec 2`), then one can expect errors of about 0.1-0.2 GPa and agreement with the total energy minimum within 0.01 bohr.

It can be activated in `run.lapw` using the `-str 0.x` option. This will change RELA to NREL in `case.struct` (and issue a warning), change PBE, PBESOL or WC to the corresponding LIBXC-options and set the STR label (instead of TOT) in `case.in0`. Once convergence has been reached, it will switch also to STR in `case.in2` and the rather expensive valence corrections will be calculated too. Note, that `lapw2` may run about 100 times slower than in TOT (FOR) mode and parallelization is highly recommended.

From the resulting tensor (`case.scf`) you can then estimate if you should increase or decrease a certain lattice parameter. Usually, the stress varies quite linear with the cell parameters and only a few steps should suffice to find the minimum (stress below 0.5 GPa). Because of the long time in `lapw2`, we recommend to use `run.lapw -I -str 0.1`, so that the expensive part is calculated only in the last scf-cycle.

**LIMITATIONS:** At present spin-polarized calculations are only possible in LDA (The spin-polarized GGA correction has still a bug). In our experience, the pressure (:PRESS in `case.scf`) is fairly robust and reliable. Unfortunately, the individual stress tensor elements (:STRESS.GPa00x) show in some cases a huge RMT dependency and sometimes seem to be quite wrong. Please always check against total energies, for instance by a variation of c/a vs. energy and compare the corresponding  $\sigma_{11}$  and  $\sigma_{33}$  values.

## Package optimize

The auxiliary program **optimize** (**x optimize [-job run-command -save savename]**) generates from an existing **case.struct** (or **case.initial.struct**, which is generated at the first call of **optimize**) a series of struct files with various volumes (or c/a ratios, or other modified parameters) (depending on your input):

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


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

```
#!/bin/csh -f
foreach i ( \
    tic_vol_-10.0 \
    tic_vol_-5.0 \
    tic_vol_0.0 \
    tic_vol_5.0 \
    tic_vol_10.0 \
)
cp $i.struct tic.struct
# cp $i.clmsum tic.clmsum
# x dstart
# run_lapw -ec 0.0001 -inlnew 3 -renorm
run_lapw -ec 0.0001
set stat = $status
if ($stat) then
    echo "ERROR status in" $i
    exit 1
endif
save_lapw ${i}
# save_lapw -f -d XXX $i
end
```

You may modify this script according to your needs: use **runsp\_lapw** or even **min\_lapw**, or specify different convergence parameters; modify the **save\_lapw** command and change the save-name or save into a directory to separate e.g. "gga" and "lda" results. Optionally 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 **epplot\_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.

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

Alternatively you can also use **epplot\_lapw** directly and the **case.analysis** file is generated automatically :

```
eplot_lapw -a vol
```

or

```
eplot -a "*" will analyse all scf files '*vol*.scf'
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 8.6 and IRelast in sec 8.9.

#### Package `optimize_abc_lapw`

The script `optimize_abc_lapw` allows for a quite efficient (minimum computational effort) optimization of lattice parameters for hexagonal, tetragonal (2D) and orthorhombic (3D) cases. However, contrary to other packages in this section, it does not give you the Bulk modulus or how eg.  $c/a$  changes with volume, but only the equilibrium lattice parameters. You can get information about its parameters using the `-h` switch:

```
optimize_abc [-h -t 2/3 -sp -p -n X -FC X -d X -ctest X Y Z -ana X
              -j "run_lapw -p ..." ]
  optimizes a, (b), c lattice parameters
  -t 2/3  2D (hexagonal, tetragonal) or 3D (orthorhombic) (default 2D)
  -sp      spinpolarized case
  -p       requires the presence of .machines (single jobstep) and
           .machines_1...4(9) for 4(9) parallel jobsteps in 2D(3D) case
  -n X     performs X optimization steps (default 5)
  -d X     delta-a (in percent) for changes in lat.params. Default 3
  -ctest  (X Y Z) stops when lat.params are converged to (X Y Z).
           Default: (0.02 0.02 0.02 bohr)
  -ana X   max number of steps for prediction of next step. Default: 3
  -FC X    convergence criterium in case.inM for -min. Default: 0.5
  -j "job" job could be a modification of the default:
           "run_lapw -I -fc 1. -p -min" (or "min_lapw ..")
```

It should be started in a directory with an initialized calculation (`init_lapw -b ...`) and it performs scf calculations (as defined by `-j "run_lapw -p -min ..."`) for the original structure and 3% changes (either  $\pm 3$  or  $+ 3$  and  $+ 6\%$ , depending on the resulting energies; can be changed with `-d X`) of  $a$ ,  $(b)$  and  $c$  (using an auxiliary script `xyzchange_lapw`), followed by a 2D (3D) parabolic fit to find the best  $a$ ,  $(b)$ ,  $c$ . This constitutes **step.1**. Since these parabolic fits are rather bad far from the minimum, it will use the newly generated structure and repeat the procedure 5 times (can be changed by `-n X`) or until the change of lattice parameters between 2 steps is below 0.02 bohr (can be changed by `-ctest X Y Z`). All the results are stored as `step.[1-X]`

files, while intermediate results are in `optimize_abc_save/[1-X]`. Debugging output is stored in `optimize_abc.out`, `xchange.out`, ...

At the end it makes automatically a `parabolfit` (see Sect.5.10.3) using:

```
parabolfit.lapw -t 2/3 -scf 'optimize_abc_save/[1-9]/*/*scf'
```

and gives a summary of the energies and lattice parameters of all steps (sometimes the lowest energy is not the last step) in `optimize_abc.summary.out`. When the starting point was very bad, it is better to repeat the `parabolfit.lapw` step with a different (less) number of steps (use `parabolfit` as given above but with a restricted range). If the lattice parameters are not converged you may add one (or more) additional steps by repeating `optimize_abc -n 1 ...`, which will create `step_6`, ... and see how the results change.

If you want to start a new optimization (eg. for a different DFT approximation) rename the `optimize_abc_save` directory before starting the new optimization.

### 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, Jamal and Reshak, 2018]) 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

```
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

```
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 `answscf=no` and a new `savename` (eg. `''_pbe_rk8_1000k''`).

### 5.3.2 Minimization of internal parameters

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

- ▶ 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.
- ▶ Alternatively, you can use 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. It usually is slower than the new method.

We recommend the first option, because this scheme is usually more efficient.

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

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

Now you have to decide which method to use:

- ▶ The new recommended way is to use a fused loop of the scf cycle and the position optimization [Marks and Luke, 2008, Marks, 2013, Marks, 2021]. You start the scf cycle with the `-min` switch. `save_lapw xxx` the original calculation and then continue with `run_lapw -min -fc 0.5 -ec 0.0001 -cc 0.001 [-it]`. This will change `case.inm` and put `MSR1a` (or `MSEC1a`) as “mixing method”. Then 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. Using the control files `-minstop` or `.minstart` (generated eg. using `touch .minstop` you can switch off/on the position optimization while an scf cycle is running. In bad cases, the final forces may not be as small as desired and possibly you have to restart this step using

MSR1a again. When having troubles we recommend you read carefully the latest \$WIEN-ROOT/SRC\_mixer/Docs. 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.

- ▶ **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).

The following text refers (mainly) to the second method using **min.lapw**:

When **case.scf** is not present, an scf-cycle will be performed first, otherwise the corresponding forces are extracted into **case.finM** and the program **mini** generates a new **case.struct** with modified atomic positions. The previous step is saved under **case\_1/2/3 . . .**. Then a new scf-cycle is executed and this loop continues until convergence (default: forces below 2mRy/bohr) is reached.

The last iteration of each geometry step is appended to **case.scf.mini**, so that this file contains the complete history of the minimization and can be used to monitor the progress (grep :ENE \*mini; or :FORxxx ...).

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

**mini** requires an input file **case.inM** (see Sec. 8.17) 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 (DELTA's, 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/au<sup>2</sup> 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 9.18). (In addition there is a program **eigenhess**, which will analyze the Hessian after the minimization has been completed. It also prints vibrational frequencies and may give you hints about dynamical instability of your system. Some more description is given in \$WIENROOT/SRC\_pairhess/README and at the top of the output file **case.outputeig**.

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

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

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

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

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

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

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

Good, since the total energy is decreasing.

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

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

When you detect 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.tmpM1 to case.tmpM
-h/-H ->       help
-NI ->         without initialization of minimization (eg. continue after a crash)
-i NUMBER ->   max. NUMBER (50) of structure changes
-s NUMBER ->   save_lapw after NUMBER of structure changes

CONTROL FILES:
.minstop        stop after next structure change

```

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

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

## 5.4 Phonon calculations

Calculations of phonons is based on a program PHONON by K.Parlinski, which must be ordered separately (see <http://www.computingformaterials.com>). Alternatively we recommend the package PHONOPY by Atsushi Togo (see [http://www.wien2k.at/reg\\_user/unsupported/](http://www.wien2k.at/reg_user/unsupported/)), which is free and has direct WIEN2k support (see examples in our workshop exercises).

### 5.4.1 PHONON

You would define the structure of your compound in PHONON together with a supercell of sufficient size (e.g. 64 atoms). PHONON will then generate a list of necessary displacements of the individual atoms. The resulting file **case.d45** must be 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.

#### init\_phonon\_lapw

**init\_phonon\_lapw** uses **case.d45** from PHONON and creates subdirectories **case.XX** and **case.XX.struct** files for all required displacements. It allows you to define globally RMT values for the different atoms and

- initializes every case individually (batch option of init\_lapw is now supported) or
- initializes every second case (useful for pos. and neg. displacements, which have the same symmetry and thus only one initialization is necessary), or
- initializes only the first case and copies the files from the first case to all others. This is most convenient in low symmetry cases with P1 symmetry for all cases and thus just one init\_lapw needs

to be executed (while for higher symmetry a separate initialization is required (but computational effort is reduced)).

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

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

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

### 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 transferred back to MS-Windows and imported into PHONON.

## 5.4.2 PHONOPY

The following example demonstrates how one can calculate the phonons in Si:

```
cd ~/WIEN2k; mkdir Si-phonon; cd Si-phonon
makestruct (Si, F cell, a=b=c=5.43 Ang, angles=90°;
           Si: (.125,.125,.125);(.875,.875,.875); setrmt 3 )
x supercell (init.struct, 1x1x1, P-lattice) # phonopy can handle only P
cp init_super.struct Si-phonon.struct
edit Si-phonon.struct # label all atoms as Si1,2,3,4
init_lapw -prec 2n # medium precision selected
phonopy --wien2k -c Si-phonon.struct -d --dim=\2 2 2"
mkdir 1; cp Si-phonon.structS-001 1/1.struct; cd 1
init_lapw -prec 2n
run_lapw -fc 0.02 # optionally use some parallelization
cp 1.scf ../; cd ..
phonopy --wien2k -f 1.scf
create band.conf with editor, containing the following information:
  ATOM_NAME = Si
  DIM = 2 2 2
  PRIMITIVE_AXIS = 0.5 0.5 0.0 0.0 0.5 0.5 0.5 0.0 0.5
  BAND = 0.5 0.5 0.5 0 0 0 0.5 0 0 0.5 0.5 0 0 0 0
  BAND_LABELS = L G X K G
  BAND_CONNECTION = .TRUE.
phonopy --wien2k -c Si-phonon.struct band.conf {p
```

In **band.yaml** you can find all phonon frequencies. Please check the PHONOPY manual for more options.

## 5.5 Running programs in parallel mode

This section describes three methods for running **WIEN2k** on parallel computers. These methods can also be combined and multiple parallelization strategies are recommended depending on the hardware and the specific case (size).

The first method uses OpenMP and is thus restricted to a single shared-memory (multicore) cpu. It is at least for smaller problems (unit cells up to 50 atoms) the first choice to use and should work even on a single laptop with a multicore cpu. The parallelization depends on a variable **OMP\_NUM\_THREADS**, which can (should) be set in your **.bashrc** or **.cshrc** file. Alternatively, you can use a **omp\_XXX** directive in **.machines** (see below). Note, that the diagonalization (often the major cpu-usage) using BLAS/LAPACK (eg. using the **mk1** or the **openblas** libraries parallelizes excellent on 2 cores, very good on 4 cores, but for more than 8 cores you may not get much improvement.

The second 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. A coarse grained 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. However, the method has some startup-delays (seconds) and limitations due to I/O overhead, so use it only AFTER OpenMP parallelization when you have more computers (nodes) available. If **lapw1** takes one hour for 10 k-points on a single node, it will scale nearly perfect up to 10 nodes. However, if **lapw1** takes only 3 seconds, the parallelization on 10 nodes may take 10 seconds instead.

The third parallelization method is based on fine grained methods, MPI, ScaLAPACK, FFTW and (optionally, but highly recommended) 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 big shared memory machine (at least 16 cores). Although for small systems (less than 50 atoms/cell) it is not as efficient as the simple OpenMP + 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  $\frac{NMAT}{\sqrt{processors}} \geq 1500$  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 all parallelization methods is possible (always use OpenMP and k-point parallelism first if you have more than 1 k-point).

### 5.5.1 k-Point Parallelization

Parts of the code are executed in k-parallel, namely **lapw1**, **lapwso**, **hf**, **lapw2**, **lapwdm** and **optic**, **qt1**, **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:** This is not necessary 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 remote login must be possible to all machines **without** specifying a password. The command for launching a remote shell can be configured during installation with **siteconfig\_lapw** (see chapter 11). Usually it is ‘**ssh**’. **ssh** must be possible to all machines **without** specifying a password. You must use public/private keys for **ssh** login. This can be done by running “**ssh-keygen -t rsa**” and pasting the **id\_rsa.pub** key into **\$HOME/.ssh/authorized\_keys** at the remote sites.

In addition, on some Linux versions, ssh will not transfer the “environment”. In this case add lines like:

```
SendEnv * # in /etc/ssh/ssh_config
AcceptEnv * # in /etc/ssh/sshd_config
```

## 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\_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<sup>1</sup>.

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[c]\_mpi**, **lapw2[c]\_mpi**, **lapwso\_mpi**, **dstart\_mpi**, **hf[c]\_mpi**, **nmr[c]\_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 k- or mpi-parallel, a switch must be set and an input file has to be prepared by the user. OpenMP parallelization works automatically if the code is compiled with OpenMP and the variable **OMP\_NUM\_THREADS** is set (or **omp\_XXX** lines are present in **.machines**, see below).

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

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

Generation of all necessary files, starting of the processes and summation of the results is done by the appropriate scripts **lapw1para**, **lapwsopara**, **hfpara**, **nlvdwpara** **lapwdmpara** and

<sup>1</sup><http://www.mpich.org/>, <http://www.netlib.org/scalapack>, <http://elpa.mpcdf.mpg.de/>, <http://www.fftw.org/>

**lapw2para** (when using **-p**), and parallel programs **dstart mpi**, **lapw0 mpi**, **lapw1 mpi**, **lapwso mpi**, **hf mpi**, **nlvdw mpi**, and **lapw2 mpi** (when using fine grained parallelization has been selected in the **.machines** file).

### 5.5.4 The **.machines** file

Besides a global OpenMP parallelization using **OMP\_NUM\_THREADS**, you can tailor the parallelization for a specific case using the following **.machines** file (assuming you have two 8 core shared memory machines and 4 (or a multiple of 4) k-points):

```
omp_global:8 # use 8 cores as default for all programs
omp_lapw1:4 # use 4 cores for lapw1
omp_lapw2:4 # use 4 cores for lapw2
1:node1
1:node2
1:node1
1:node2 # use k-parallelization with 4 jobs on 2 nodes
```

**lapw0** would run on 8 cores, but **lapw1/2** would use only 4 cores, but 4 jobs would be issued in parallel, each one with  $1/4^{th}$  of the k-points.

Other possible OpenMP keywords for different programs are listed in **\$WIENROOT/SRC.templates/.machines**.

The following **.machines** file describes a more complicated example (assuming no OpenMP parallelization) combining k-point and mpi parallelization. We assume to have 5 computers, (alpha, ... epsilon), all have 4 cores, but only alpha has enough memory to run your case on a single node and is twice as fast as the others. Fortunately, beta ... epsilon have a fast network and the smaller memory nodes can be combined using mpi.

```
granularity:1
1:alpha
4:beta:4 gamma:4 delta:4 epsilon:4
residue:alpha
balance:
lapw0:beta:4 gamma:4 delta:4 epsilon:4
dstart:beta:4 gamma:4 delta:4 epsilon:4
nlvdw:beta:4 gamma:4 delta:4 epsilon:4
```

To each set of processors, defined by a single line starting with a “weight:” 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 10 k-points to be distributed in the above example, they are distributed as follows. The computer **alpha** gets 2 k-points. The other 8 k-points will be run in a second job in



mpi-parallel mode on 16 (slower) cores. You should then check the timing in `case.dayfile` and possibly adjust the “speed” (for instance to 3 and 7).

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 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 -qt1`”) 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  $\text{MOD}(klist, \sum_j \text{newweight}_j)$ ) 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
```

define 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. One 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:

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

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

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

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

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

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

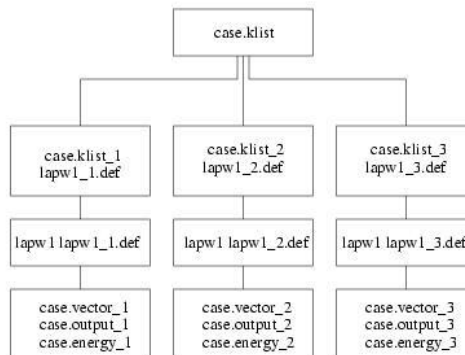
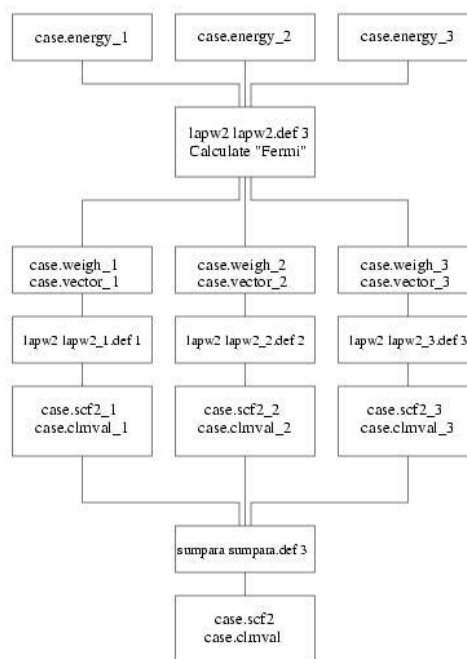
In **WIEN2k** usually only a small part of time is spent in the programs `lapw0`, `lcore` and `mixer` which is very small (negligible) in comparison to the times spent in `lapw1` and `lapw2`. The time for waiting until all parallel `lapw1` and `lapw2` processes have finished is important too. For a good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

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

### 5.5.6 Flow chart of the parallel scripts

To see how files are handled by the scripts `lapw1para` and `lapw2para` refer to figures 5.1 and 5.2. After the `lapw2` calculations are completed the densities and the information from the `case.scf.x` files are summarized by `sumpara`.

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

Figure 5.1: Flow chart of **lapw1para**Figure 5.2: Flow chart of **lapw2para**

### 5.5.7 On the fine grained parallelization

The following parallel programs use different parallelization strategies:

**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, 16, ... processors), if possible, give best performance due to ScaLAPACK. However, other grids are also possible (e.g. 4x2=8, 5x4=20, 8x4=32, ...). The default setup for rectangular 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. 7.6.3).

OpenMp, k-point and mpi-parallelization can be used at the same time and are specified by the lines “speed:hostname:number\_of\_cores” and the “omp-directives” in **.machines**.

**hf\_mpi** If you use the **-hf** option for **run\_lapw** (“full hybrid” scheme, see Sect. 4.5.9, or the Slater/SmBJ/KLI potential, see Sect. 4.5.10), 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 (**-mode1** or **-mode2**, see details in Sect. 4.5.9). 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\_mpi** 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\_mpi** 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  $\sqrt{N}/4$ . The parallelization follows that of **lapw1** as specified in **.machines**, although the script uses **.processes**, which has been created in the **lapw1** step.

**lapw2\_mpi** is parallelized in two main parts: (i) The density inside the spheres is parallelized over atoms, and (ii) the fast Fourier transforms are done in parallel.

In addition the density calculation for each atom can be further parallelized by distributing

the eigenvector on a certain subset of processors (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.mpi** 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 extent. Matrix setup in **lapw1** should scale nearly perfect, while diagonalization (using ScaLAPACK) will not. Please note: When running the “TiC”-example (Quick start) in mpi-mode on 16 cores, it will be MUCH slower than on a single core in sequential mode. ALWAYS check the actual speed-up when increasing the number of cores. Usually, “iterative” scales better than “full” diagonalization and is preferred for large scale computations (and surfaces). 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  $\pm x, y, z$  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 8.19) 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 `case.in1.nmr` file (`case.in1c.nmr` 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). Note: HDLOs in `case.in1` are not supported and must be removed.

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`)

```
x.nmr.lapw -mode lapw1
```

In a case where spin-orbit coupling needs to be included ("**x.nmr -so**") the proper eigenvectors are generated with:

```
x.nmr.lapw -mode lapwso
```

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/case_A
```

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 shielding  $\sigma_{iso}$  and its anisotropy is printed under the label ":NMR-TOTxxx" (in ppm) and :NMRASYxxx (Haerberlen convention):

```
:NMRTOT001 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.

## 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 in1
              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
  -ovlpmx val  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 lapw1
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_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)

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
-iemin val     sets lowest valence band to val
-iemax val     sets highest valence band to val
-filt_cxyz_o iat l  filter coupling matrix element (<OS|COUPOP|ES>,make_cxyz)
                 the occupied states <OS|. Leaves only nonzero alm for iat
                 and l (|FOP_oc>=SUM_es(|ES><ES|COUPOP|OS>/(ENE_os-ENE_es)
-filt_cxyz_q iat l  filter in coupling matrix elements (<OS|COUPOP|ES>,make_cxyz)
                 the empty states |ES>. Leaves only nonzero alm for iat
                 and l (|FOP_oc>=SUM_es(|ES><ES|COUPOP|OS>/(ENE_os-ENE_es)
-filt_curr_o iat l  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 l  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
-up/dn        include spin polarization (up/dn spin)
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.

**NMR and hybrid DFT :**

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 **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:

```
x_nmr_lapw -p -hf
```

or, if you have enough cores create several different **.machines** files and run the 7 directories in parallel:

```
cp .machine_q0 .machines
x_nmr_lapw -p -hfdir q0 &
cp .machine_pqx .machines
x_nmr_lapw -p -hfdir pqx &
...
```

**Analyses :**

```
x_nmr_lapw -p -noinit -emin xx [-emax yy]
```

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 **-noinit** runs only the modes current and integ. Additional analysis is possible with the **-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]).

**Current plotting :**

You can also plot the induced current (it needs the **dx** Dataexplorer software), but since the current is not symmetrized, you need to run first with a full k-mesh. Use

```
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
```

**Summary :**

If you have many atoms in your cell (eg. in supercells), you can get an automatic summary of your shifts in **case.outputnmr.integ** using:

```
nmr_orb_analyse case 1 2 3 ... # specify as many atoms as you need
```

It produces **summary.nmr\_orb**.

**5.6.2 Knight shifts**

For paramagnetic metals, the **Knight shift** dominates usually the **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 a subdirectory of the non-spinpolarized calculation (usually called **spin**) using a spin-polarized setup and a magnetic field applied self-consistently in **lapw0**. Execute :

```
mkdir spin; cp case.struct spin/spin.struct; cd spin
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 -cc 0.00001 [-p] ... # run scf with zero moment
save_lapw zero_moment
```

```

cp $WIENROOT/SRC_templates/case.vorbup.100T spin.vorbup # contains
the 100 T field
cp $WIENROOT/SRC_templates/case.vorbdn.100T spin.vorbdn
runsp_lapw -orbcc -cc 0.000001 [-p] ... # scf calculation with field
grep line :HFF0XX spin.scf # get the hyperfine field for atom XX
save_lapw 100T_XXX-kpoints_fermit-4 # save the calculation

```

**contact term :**

The contact term of the shielding can be obtained from the HFF vales in the scf file (in kGauss) as:  $\sigma_c[ppm] = -HFF * 1000$  (for a 100 T field, mind the minus sign).

**spin susceptibility :**

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

$$\chi_s[cm^3 mol^{-1} cell^{-1}] = \frac{M[\mu_B]}{B[T]} * 0.5584939 \quad (5.2)$$

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

**k-mesh convergence :**

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).

**dipolar term :**

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=1-3 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  $\sigma_{sd}[ppm]$  by multiplication with -10000 (for a field of 100 T, mind the minus sign).

**orbital term for metals :**


Of course, also the orbital contribution is non-negligible in metallic compounds and one should also calculate the chemical shielding (see above). Run **x.nmr -metal [-p] [-nox] [-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 -nox 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**.

**summing up all contributions :**

To add up all the contributions, you should first use **nmr orb analyse** to obtain **summary\_nmr\_orb** (in the parent directory of your spin-polarized calculation). Then use **nmr analyse case 1 2 3 ...** to add the orbital + contact + (optional) dipolar terms. The summary is in **summary\_nmr**.

## 5.7 Wannier functions (`wien2wannier`)

This program was contributed by:

Wien2Wannier by J.Kunes, P.Wisgott and E.Assmann. Please cite the following paper when using it:  
 J.Kunes, R.Arita, P.Wisgott, A.Toschi, H.Ikeda, K.Held, Comp.Phys.Comm. 181, 1888 (2010)  
<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 auxiliary programs it also contains a package for plotting the resulting Wannier functions 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's` “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  $[E_{min}; E_{max}]$  (in eV with EF=0), and outputs the corresponding band indices  $b_{min}; b_{max}$ . 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  $b_{min}; b_{max}$  have to be specified, and initial projections  $A_{mn}$  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.nknp**.
- ▶ **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] [-hf]: 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 \\  
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**.
- ▶ **xcrysdem --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, Weyl points (BerryPI)

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, tutorials and updates are also available from: <https://github.com/rubel175/BerryPI>  
 email: [rubelo@mcmaster.ca](mailto:rubelo@mcmaster.ca)  
 Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

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 **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 ([Ahmed et al., 2013]), which should be cited when this module is used in a publication) or the detailed tutorials at **\$WIENROOT/SRC\_BerryPI/BerryPI**.

We strongly suggest that you read and repeat the tutorials as described on the wiki page at <https://github.com/rubel175/BerryPI>.

As this method should be applied to insulators only, you have to use the TETRA method for the BZ integration.

### 5.8.1 Options

The program is called using

```
berrypi [ -k NX NY NZ -sp -orb -so --skip-lapw -p -w -b NUM1  
NUM2 -sp_c] [-h]
```

An online help of all options can be obtained with the **-h** switch.

The parameter **-k NX NY NZ** determines the k-mesh for the BZ sampling (default: 4 4 4).

The additional switches **-sp** allows spin-polarized calculations; **-orb** supports additional orbital potentials (LDA+U or EECE); **-so** includes spin-orbit coupling (**x lapwso**); and **--skip-lapw** skips the **lapw1** run and uses a previous **case.vector** file. The option **-p** can be used to make

## 5.8. SPONTANEOUS POLARIZATION, PIEZOELECTRICITY AND BORN CHARGES, WEYL POINTS (BERRYPI)11

the most time consuming `lapw1` step running in parallel (a `.machines` file is necessary). The `-w` option computes the Berry phase along a specific k-path (a closed loop) given in the case.klist file. It can only be done in calculations with spin-orbit coupling and should be coupled with `-b NUM1 NUM2` to use a specified band range only. It is used to characterize Weyl-points.

### 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<sub>3</sub>) and perform a standard WIEN2k calculation. Then run the berrypi program:

```
mkdir case;cd case;mkdir case0;mkdir case1      # create suitable directories
cd case1
makestruct                                     # create your structure
init_lapw -b ...                               # initialize wien2k
run_lapw ...                                   # run scf cycle
berrypi -k 6 6 6                              # run berrypi
```

where `-k` defines a suitable k-mesh. This will give you the corresponding x,y,z components of the polarization in primitive and conventional lattice coordinates, respectively. Note, the polarization is defined only with respect to a phase factor and thus the computed values are given with both,  $\pi$ -wrapping from  $-\pi$  to  $+\pi$  and 0 to  $2\pi$ :

```
=====
Value | spin | dir(1) | dir(2) | dir(3)
-----|-----|-----|-----|-----
Electronic polarization (C/m2) sp(1) [-9.684673e-12, -2.406503e-13, 4.879618e-01]
Ionic polarization (C/m2) sp(1) [ 1.365657e-11, 1.365657e-11, -1.760570e-01]
Tot.spin polariz.=Pion+Pel(C/m2) sp(1) [ 3.971897e-12, 1.341592e-11, 3.119048e-01]
-----|-----|-----|-----|-----
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 -k 6 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/m<sup>2</sup>. Please consider the effects of possible  $\pi$  wrapping, so in general the smallest possible value should be considered. If there is a suspect of  $\pi$ -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 we create first a “P”-type supercell with 4 formula units/cell (this is not necessary anymore with versions starting from 2021, since all lattice types are now supported by BerryPi). One of the 4 As atoms has to be displaced along the z-axis from its 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 -k 6 6 6**. The two calculations yield lines like:

```
"lambda1"
ELECTRONIC POLARIZATION
=====
Value          | spin | dir(1) | dir(2) | dir(3)
-----
...
Berry phase (rad) [-pi ... +pi] up+dn [ 3.151667e-10, -2.544453e-09, -1.081339e+00]
Electronic polarization (C/m2) sp(1) [ 2.441627e-11, -1.971212e-10, -8.377237e-02]
=====
IONIC POLARIZATION
=====
Elem. | Fractional coord. | spin |val| dir(1) | dir(2) | dir(3)
-----
Total ionic phase wrap. (rad) sp(1) [ 3.686359e-09, 3.686359e-09, 9.424778e-01]
Ionic polarization (C/m2) sp(1) [ 2.855857e-10, 2.855857e-10, 7.301465e-02]
=====

"lambda2"
ELECTRONIC POLARIZATION
=====
Value          | spin | dir(1) | dir(2) | dir(3)
-----
...
Berry phase (rad) [-pi ... +pi] up+dn [ 7.675118e-10, -2.577606e-09, 1.081339e+00]
Electronic polarization (C/m2) sp(1) [ 5.945987e-11, -1.996896e-10, 8.377237e-02]
=====
IONIC POLARIZATION
=====
Elem. | Fractional coord. | spin |val| dir(1) | dir(2) | dir(3)
-----
Total ionic phase wrap. (rad) sp(1) [ 3.686359e-09, 3.686359e-09, -9.424778e-01]
Ionic polarization (C/m2) sp(1) [ 2.855857e-10, 2.855857e-10, -7.301465e-02]
=====
```

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

$$Z_{zz}^* = \frac{1}{2\pi} \frac{\delta\phi_z}{\delta\rho_z} \quad (5.3)$$

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  $\text{PbTiO}_3$  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:

$$\epsilon_{zz} = \frac{dP_z}{d\epsilon_z} \quad (5.4)$$

### 5.8.5 Weyl points

The characterization of topological properties and in particular of Weyl points helps to determine important transport properties in modern materials. The method is based on Wilson loops (loops in k-space around a Weyl point) and characterizes the chirality of such points (see [Saini et al., 2022]). For specific examples how to characterize such properties, see the corresponding tutorials at <https://github.com/rubel75/BerryPI/wiki/Tutorial-5:-Weyl-points-characterization-in-TaAs> and <https://github.com/rubel75/BerryPI/wiki/Tutorial-6:-Weyl-point-characterization-in-Te>.

## 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 \$PDF-READER). You can search for a specific keyword using “**^f keyword**”. This procedure substitutes an “Index” and should make it possible to find a specific information without reading through the complete users guide.

- ▶ In addition there is a html-version of the UG and its starting page is:  
**\$WIENROOT/SRC\_usersguide\_html/usersguide.html**
- ▶ When using the user interface **w2web**, you have access to the html and pdf-version (the latter requires an X-windows environment) of the usersguide.
- ▶ At our webserver [http://www.wien2k.at/reg\\_user](http://www.wien2k.at/reg_user) we put information 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_lapw`

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 `grep_line` (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.analysis` automatically using the specified string.

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

```
eplot_lapw -h
```

### 5.10.2 `gibbs_lapw`

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:

```
cd dir1
x optimize # select Volume optimization and a suitable volume range
optimize.job # optionally 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 # optionally 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`. The energy/volume data are read from `case.analysis`, except in case of the `-2D` switch, where it reads from `case.2DEOS`.

### 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

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

you can optionally specify the dimensionality of the fit, the specific scf-filenames or which angle ( $\alpha/\beta/\gamma$ ) should be analysed.

### 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 (from `case.dosX`) or (after running `x pes`) the total and partial photoelectron spectra (PES) in `case.pesX`, or the broadened PES spectra after `pes` and `broadening` (`case.pesb`, or the renormalized DOS (after `pes` or `rendos` in `case.dosrnXev`) depending on the input provided by `case.int` and the interactive input. It can plot up to 4 DOS/PES 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

```
dosplot2_lapw [-up|-dn -ren -pes -pesb -i -layout -h]
```

The switch `-i` ignores the presence of a startup file (`dosplot.ini`), `-layout` will allow to specify details of the plots (color, line type,...), `-pes` and `-pesb` plots the (broadened) photoelectron spectra, `-ren` the renormalized DOS (both calculated using the `pes` or the `rendos` program).

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**.*

### 5.10.10 prepare\_xsf\_lapw

This program was contributed by:



David Koller  
Institute for Materials Chemistry  
TU Vienna

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 XCrysDen-format (`case.xsf`) for plotting with `XCrysDen` (Menu → Tools → Data Grid). It is written in Python and also uses the programs `lapw5` and `str2xcr.exe` (included in the WIEN2k distribution). It is rather slow and should be replaced by `3ddens`, which is orders of magnitude faster.

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 0.5 0.1 0 # x-end
>CY 0.1 0.5 0 # y-end
>CZ 0.2 0.2 1 # z-end
# use for fcc:
#>C0 0 0 0
#>CX 0.5 0.5 0
#>CY 0.5 0 0.5
#>CZ 0 0.5 0.5
# entire cell:
#>C0 0 0 0
#>CX 1 0 0
#>CY 0 1 0
#>CZ 0 0 1

>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 ( $\epsilon_1$ ) 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, ....

### 5.10.13 create\_elf\_lapw

The script `create_elf_lapw` generates the file `case.rho.onedim` (1D plot), `case.rho` (2D plot) or `case.xsf` (3D plot) for the plotting of the electron localization function (ELF)

$$\text{ELF} = 1 / \left( 1 + \left( (\tau - \tau^W) / \tau^{TF} \right)^2 \right) \quad (5.5)$$

the iso-orbital indicator (used in meta-GGAs)  $\alpha$

$$\alpha = (\tau - \tau^W) / \tau^{TF} \quad (5.6)$$

or

$$z = \tau^W / \tau, \quad (5.7)$$

with  $\tau, \tau^W = |\nabla\rho|^2 / 8\rho$  and  $\tau^{TF} = (3/19)(3\pi^2)^{2/3} / 3\rho^{5/3}$  being the true Kohn-Sham, von Weizsäcker and Thomas-Fermi kinetic energy density.

**create\_elf\_lapw** executes **lapw0** and then **lapw5** (for a 1D and 2D plot) or **3ddens** (for a 3D plot) for  $\tau$ ,  $\tau^{TF}$  and  $\tau^W$ , individually. Then, **create\_elf\_lapw** executes **create\_rho**, which calculates ELF,  $\alpha$  or  $z$  and stores it in **case.rho.onedim/rho/xsf**.

The advantage of using **create\_elf\_lapw** instead of directly the keyword VX\_ELF, VX\_ALPHA and VX\_Z in **case.in0** (see Table 7.6) is to have a function that is better converged in the interstitial region, especially for  $\alpha$  (typically, a badly converged Fourier series displays oscillations).

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

```
create_elf_lapw -h
```





---

# 6 Programs for the initialization

---

## Contents

---

6.1	NN	125
6.2	SGROUP	126
6.3	SYMMETRY	126
6.4	LSTART	127
6.5	KGEN	130
6.6	DSTART	130

---

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) and bond angles for the first coordination sphere are written to **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 [-add]
```

The switch `-add` calculates BVA also for H- or C-C and N-N bonds.

## 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 DFT 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 `R0` is too big or core-density leaks out of RMT.

### 6.4.1 Execution

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

```
lstart lstart.def or x lstart [-sigma -half -tau -hdlo]
```

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).

Using `-half` takes an alternative input `case.inst.half`, and generates alternative output files (additional `.half` suffix) for the DFT-half method (see section 4.5.14).

Using `-tau` activates output of the atomic kinetic energy density (slater form  $\nabla\psi^* \cdot \nabla\psi$ ), which may be used to start a self-consistent gKS MGGA calculation, or an mBJ calculation, from a superposition of atomic (kinetic energy) densities.

Using `-hdlo` directs `lstart` to create HDLOs for p, d or f states (provided there is no corresponding semicore) in `case.in1`.

## 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,OCCUP; = 3S UP, 1 ELECTRON)
3,-1,1.0 N    3S DN
3,-2,2.0 N    3P UP
3,-2,2.0 N    3P DN
3, 1,1.0 N    3P*UP
3, 1,1.0 N    3P*DN
3,-3,3.0 P    3D UP
3,-3,3.0 P    3D DN
3, 2,2.0 P    3D*UP
3, 2,2.0 P    3D*DN
4,-1,1.0 P    4S UP
4,-1,1.0 P    4S DN
****          END OF Input
****          END OF Input
----- bottom of file -----

```

Interpretive comments follow:

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

title

keyword

The keyword **Watson** enables a stabilization of negative ions using a “Watson”-sphere of radius `R-wat` with charge `Q-wat`, which must be given in the next line when this keyword is specified.

The keyword **PRATT** enables a scf mixing using standard PRATT scheme. It might be 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. [Lieberman et al., 1965]):

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

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

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

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

Table 6.6: Relativistic quantum numbers

## 6.5 KGEN (generates k mesh)

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

**kgen** needs as interactive input the total number of k-points in the full BZ.

If this number is 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) to create a mesh yourself.

If this number is -1, you should give a k-point spacing in bohr<sup>-1</sup>, i.e. specify the distance between 2 neighboring k-points (typically 0.3 to 0.05, depending on desired accuracy and metal/nonmetal character).

If inversion symmetry is not present, it will be added automatically unless you specified the "-so" switch and **case.ksym** is present (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 155.

### 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 does not add inversion symmetry. The switch **-fbz** generates a k-mesh in the full Brillouinzone (no symmetry).

### 6.5.2 Dimensioning parameters

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

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

## 6.6 DSTART (superposition of atomic densities)

This program generates an initial crystalline charge density **case.clmsum** by a superposition of atomic densities (**case.rsp**) generated with **lstart**. Information about LM values of the lattice harmonics representation and number of Fourier coefficients of the interstitial charge density are taken from **case.in1** and **case.in2**. You may also specify a larger LUSE value in **case.in0** (default LUSE=13) 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**. The program is also used for superposition of atomic potentials required for DFT-1/2 calculations (see section 4.5.14).

### 6.6.1 Execution

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

```
dstart dstart.def  or  x dstart [-up|dn -fft -super -lcore -half  
-tau -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.rsplcore** ( core-superposition, this is activated during scf when a **.lcore** file is present.

With **-half** the program runs in atomic superposition mode for DFT-1/2: **case.inpd case.pot**, and **case.pot\_half** are read and **case.r2v\_half** is produced.

It can run automatically on OpenMP parallel mode or using (**-p**) in mpi-parallel mode for big cases (typically more than 20 atoms).

With **-tau**, **dstart** performs the superposition of the kinetic energy density, which may be used to start a self-consistent gKS MGGGA calculation, or an mBJ calculation, from a superposition of atomic (kinetic energy) densities. In this case it is necessary to run **x lstart -tau** first.

You can also select a different density truncation inside spheres, when calculating the PW coefficients. Create **case.indstart** and enter a "mode" (0-3) and eventually height-factor (mode=1) or an exponent (1,2,3 for mode=3).For details check **old.dstart.F**. We did not observe a significant reduction of scf-cycles with either of the new methods, although sometimes it can give some improvement.

### 6.6.2 Dimensioning parameters

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

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





---

# 7 Programs for running an SCF cycle

---

## Contents

---

7.1	LAPW0	133
7.2	DFTD3	144
7.3	DFTD4	145
7.4	NLVDW	146
7.5	ORB	147
7.6	LAPW1	151
7.7	HF	156
7.8	LAPWSO	158
7.9	LAPW2	160
7.10	SUMPARA	165
7.11	LAPWDM	166
7.12	LCORE	167
7.13	MIXER	169

---

In sections 7.1-7.13 we describe the main programs to run an SCF cycle as illustrated in figure 4.1.

## 7.1 LAPW0 (generates potential)

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

For gKS MGGA calculations, **lapw0** is executed twice. Once to generate the auxiliary local GGA potential (used for core and radial functions), and once to generate the gKS MGGA potential (for details see section 4.5.17). In the first case, the input file **case.in0\_loc\_vsp** is used, and only the file **case.vsp (up/dn)** is used as output. In the latter case, the usual input file **case.in0** is used, but there is additional in/output. The required input is both the density **case.clmsum/up/dn** and the kinetic energy density **case.tau (sum/up/dn)**. The additional output are the file **case.vspmgsa (up/dn)** which contains the spherical, multiplicative part of the gKS potential and the file **case.vtau (up/dn)** which contains the non-multiplicative part of the gKS potential. The file **case.vns (up/dn)** contains the non-spherical, multiplicative part of the gKS potential. If **R2V** is set in the input (see below), a file **case.r2v2** containing the non-multiplicative part of the exchange-correlation potential for plotting is written, whereas **case.r2v** contains the multiplicative part.

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). In case the pseudo density is not well converged a :WARning is issued and one should probably increase GMAX or decrease NCON.

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 Lebedev or 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  $V_{xc}$  and these potentials can also be printed into separate files (see below).

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

The contribution to the total energy which involves integrals of the form  $\rho * V$  is calculated according to the formalism of [Weinert et al., 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 (workfunction =  $v_{zero} - EF$ ; test if your vacuum is large enough; it is assumed that the middle of your vacuum region is either at  $z=0$  or  $z=0.5$ ).

### 7.1.1 Execution

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

```
lapw0 lapw0.def or x lapw0 [ -p -eece -grr -nlvdw -half]
```

### 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:  $\text{NCOM}+3 .gt. \{[\text{number of } l, m \text{ with } m = 0] + [2 * \text{number of } l, m \text{ with } m > 0]\}$   
**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:

```

----- top of file: case.in0 -----
TOT  XC_PBE          # MULT/COUL/EXCH/POT /TOT ; VXC-SWITCH
NR2V  IFFT 13       # R2V EECE/HYBR IFFT LUSE
 30 30 108 2.00 1 NCON 9 # IFFT-parameters, enhancement factor, iprint, NCON
0 0.0      (#of FK in E-field expansion, EFELD (Ry))
----- bottom of file -----

```

or, when a staggered field (section 4.5.4) is turned on:

```

----- top of file: case.in0 -----
TOT  XC_PBE          # MULT/COUL/EXCH/POT /TOT ; VXC-SWITCH
NR2V  IFFT 13       # R2V EECE/HYBR IFFT LUSE
 30 30 108 2.00 1 NCON 9 # IFFT-parameters, enhancement factor, iprint, NCON
STAGFIELD
2      natom
1 -0.01  iatom, shift for spin up (Ry)
2  0.01  iatom, shift for spin up (Ry)
----- bottom of file -----

```

Interpretive comments follow:

#### line 1: free format

switch, indxc, xc1, xc2, xc3

#### switch

**TOT**    total energy contributions and total potential calculated  
**STR**    stress and 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

NOTE: MULT, COUL, and EXCH are for testing only, whereas POT, saves some CPU time if total energy is not needed

**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 4.5.8 and 4.5.9, 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.tau\*** files, you need **case.inm.tau** (**cp \$WIENROOT/SRC\_templates/template.inm.tau case.inm.tau** and run one scf cycle with XC.PBE after creation of **case.inm.tau**. Only afterwards change `indxc` to XC.SCAN. Meta-GGA functionals may require a larger IFFT factor (**case.in0**) or GMAX (**case.in2**) than GGA functionals.
- ▶ XC.MBJ or XC.LMBJ: modified Becke-Johnson (mBJ) potential  $V_{XC}$  [Tran and Blaha, 2009] or its local version lmBJ [Rauch et al., 2020]. Uses the (l)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 about (l)mBJ calculations see Secs. 4.5.11 and 4.5.12.

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)

XC.LDA.X\_NAME  
 XC.LDA.C\_NAME  
 XC.LDA.XC\_NAME  
 XC.GGA.X\_NAME  
 XC.GGA.C\_NAME  
 XC.GGA.XC\_NAME  
 XC.MGGA.X\_NAME  
 XC.MGGA.C\_NAME  
 XC.MGGA.XC\_NAME

Keywords to use functionals from the library of exchange and correlation functionals **LIBXC** (<https://libxc.gitlab.io/>). A few points about the use of **LIBXC**:

- ▶ **LIBXC** [Marques et al., 2012, Lehtola et al., 2018] is a separate program that has to be downloaded and compiled with the same compiler as the one used for **WIEN2k** (see 11.1.1).
- ▶ One (*\_X\_*, *\_C\_* or *\_XC\_*) or two (*\_X\_* and *\_C\_*) **LIBXC** keywords can be specified. The full list of possible keywords is in `$WIENROOT/SRC.lapw0/xc_funcs.h` or on the **LIBXC** website.
- ▶ It is possible to combine a functional/potential from **LIBXC** with a functional/potential from **WIEN2k** by also specifying one or several of the keywords *EX\_NAME*, *EC\_NAME*, *VX\_NAME*, *VC\_NAME* (but not *XC\_NAME*) which will overrule the corresponding **LIBXC** choice (except if *NAME=NONE*).
- ▶ A few **LIBXC** keywords (*XC\_GGA\_X\_LB*, *XC\_GGA\_X\_LBM*, *XC\_MGGA\_X\_BJ06*, *XC\_MGGA\_X\_TB09*, *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 *EX\_NAME* and *EC\_NAME*.
- ▶ As for the non-scf meta-GGA energy functionals called with the **WIEN2k** keywords (see table 7.3), the default associated potential is PBE.
- ▶ For stress calculations with a GGA, it is mandatory to use **LIBXC** (set automatically when using the `-str` convergence criterium in `run.lapw`).
- ▶ For scf mGGA calculations it is mandatory to use **LIBXC** (set automatically with `init.mgga`).
- ▶ Note that the possibility offered by **LIBXC** to specify the value of parameters in several functionals has not been implemented in **WIEN2k**.

Example for PBE with the

- ▶ global keyword : `XC_PBE`
- ▶ individual keywords : `EX_PBE EC_PBE VX_PBE VC_PBE`
- ▶ **LIBXC**-keywords : `XC_GGA_X_PBE XC_GGA_C_PBE`

`xc1,xc2,xc3`

optional input(s) for certain XC options:

Example for `MGGA_MS2`: `XC_MGGA_MS 0.504 0.14601 4.0`

`XC_LDA` or `XC_PBE`: to modify the spin scaling (reduction of spin-polarization) according to ([Ortenzi et al., 2012]). `xc1` must be between 0 and 2.

Global keyword	EX.SWITCH	EC.SWITCH	VX.SWITCH	VC.SWITCH	Type	Comments
XC.LDA	EX.LDA	EC.LDA	VX.LDA	VC.LDA	LDA	
XC.PBE	EX.PBE	EC.PBE	VX.PBE	VC.PBE	GGA	
XC.WC	EX.WC	EC.PBE	VX.WC	VC.PBE	GGA	
XC.PBESOL	EX.PBESOL	EC.PBESOL	VX.PBESOL	VC.PBESOL	GGA	
XC.B3PW91	EX.B3PW91		VX.B3PW91		GGA	semilocal part of hybrid B3PW91
XC.B3LYP	EX.B3LYP		VX.B3LYP		GGA	semilocal part of hybrid B3LYP
XC.MBJ	EX.LDA	EC.LDA	VX.MBJ	VC.LDA	MGGA	
XC.LMBJ	EX.LDA	EC.LDA	VX.LMBJ	VC.LDA	MGGA	
XC.TPSS	EX.TPSS	EC.TPSS	VX.PBE	VC.PBE	MGGA	
XC.REVTPSS	EX.REVTPSS	EC.REVTPSS	VX.PBE	VC.PBE	MGGA	
XC.MGGA_MS	EX.MGGA_MS	EC.MGGA_MS	VX.PBE	VC.PBE	MGGA	xc1 ( $\kappa$ ), xc2 ( $c$ ) and xc3 ( $b$ ) are required
XC.MVS	EX.MVS	EC.MVS	VX.PBE	VC.PBE	MGGA	
XC.MBEEF	EX.MBEEF	EC.PBESOL	VX.PBE	VC.PBE	MGGA	
XC.SCAN	EX.SCAN	EC.SCAN	VX.PBE	VC.PBE	MGGA	
XC.SCANL	EX.SCANL	EC.SCANL	VX.PBE	VC.PBE	MGGA	
XC.RSCAN	EX.RSCAN	EC.RSCAN	VX.PBE	VC.PBE	MGGA	
XC.R2SCAN	EX.R2SCAN	EC.R2SCAN	VX.PBE	VC.PBE	MGGA	
XC.TM	EX.TM	EC.TM	VX.PBE	VC.PBE	MGGA	

Table 7.3: global XC-switches

Keyword for $E_x$	Type	Reference	Comments
EX_NONE			no exchange energy
EX.LDA	LDA	[Kohn and Sham, 1965]	
EX.SLDA	LDA	[Tran and Blaha, 2011]	screened LDA for hybrid ( <b>case.in0.grr</b> )
EX.B88	GGA	[Becke, 1988]	
EX.PW91	GGA	[Perdew et al., 1992]	
EX.EV93	GGA	[Engel and Vosko, 1993]	
EX.PBE	GGA	[Perdew et al., 1996]	
EX.REVPBE	GGA	[Zhang and Yang, 1998]	
EX.RPBE	GGA	[Hammer et al., 1999]	
EX.HCTH93	GGA	[Hamprecht et al., 1998]	
EX.HCTH120	GGA	[Boese et al., 2000]	
EX.HCTH147	GGA	[Boese et al., 2000]	
EX.HCTH407	GGA	[Boese and Handy, 2001]	
EX.AM05	GGA	[Armiento and Mattsson, 2005]	
EX.WC	GGA	[Wu and Cohen, 2006]	
EX.PBE-ALPHA	GGA	[Madsen, 2007]	xc1 ( $\alpha$ ) is required
EX.PBESOL	GGA	[Perdew et al., 2008]	
EX.SOGGA	GGA	[Zhao and Truhlar, 2008]	
EX.RGE2	GGA	[Ruzsinszky et al., 2009]	
EX.PBEINT	GGA	[Fabiano et al., 2010]	
EX.OPTPBE	GGA	[Klimeš et al., 2010]	
EX.OPTB88	GGA	[Klimeš et al., 2010]	
EX.OPTB86B	GGA	[Klimeš et al., 2011]	
EX.HTBS	GGA	[Haas et al., 2011]	
EX.AK13	GGA	[Armiento and Kümmel, 2013]	
EX.CAP	GGA	[Carmona-Espindola et al., 2015]	
EX.SG4	GGA	[Constantin et al., 2016]	
EX.MPBE	GGA	[Haas et al., 2010]	xc1 ( $\mu$ ) and xc2 ( $\kappa$ ) are required
EX.B3PW91	GGA	[Becke, 1993]	semilocal part of hybrid B3PW91
EX.B3LYP	GGA	[Stephens et al., 1994]	semilocal part of hybrid B3LYP
EX.SPBE	GGA	[Tran and Blaha, 2011]	screened PBE for hybrid ( <b>case.in0.grr</b> )
EX.SWC	GGA	[Tran and Blaha, 2011]	screened WC for hybrid ( <b>case.in0.grr</b> )
EX.SPBESOL	GGA	[Tran and Blaha, 2011]	screened PBESol for hybrid ( <b>case.in0.grr</b> )
EX.SB88	GGA	[Tran and Blaha, 2011]	screened B88 for hybrid ( <b>case.in0.grr</b> )
EX.SHJSPBE	GGA	[Henderson et al., 2008]	screened PBE for hybrid ( <b>case.in0.grr</b> ), $\omega = (2/3)\lambda$
EX.SHJSPBESOL	GGA	[Henderson et al., 2008]	screened PBESol for hybrid ( <b>case.in0.grr</b> ), $\omega = (2/3)\lambda$
EX.SHJSB88	GGA	[Weintraub et al., 2009]	screened B88 for hybrid ( <b>case.in0.grr</b> ), $\omega = (2/3)\lambda$
EX.SHJSB97X	GGA	[Henderson et al., 2008]	screened B97x for hybrid ( <b>case.in0.grr</b> ), $\omega = (2/3)\lambda$
EX.VSXC	MGGA	[Van Voorhis and Scuseria, 1998]	
EX.PKZB	MGGA	[Perdew et al., 1999]	
EX.TPSS	MGGA	[Tao et al., 2003]	
EX.REVTPSS	MGGA	[Perdew et al., 2009]	
EX.MGGA_MS	MGGA	[Sun et al., 2013]	xc1 ( $\kappa$ ), xc2 ( $c$ ) and xc3 ( $b$ ) are required
EX.MVS	MGGA	[Sun et al., 2015a]	
EX.MBEEF	MGGA	[Wellendorff et al., 2014]	
EX.SCAN	MGGA	[Sun et al., 2015b]	
EX.SCANL	MGGA	[Mejia-Rodriguez and Trickey, 2018]	
EX.RSCAN	MGGA	[Bartók and Yates, 2019]	
EX.R2SCAN	MGGA	[Furness et al., 2020]	
EX.TM	MGGA	[Tao and Mo, 2016]	
EX.GRR		[Tran and Blaha, 2009]	average of $ \nabla\rho /\rho$ for mBJ ( <b>case.in0.grr</b> )

Table 7.4: EX-switches

Keyword for $E_C$	Type	Reference	Comments
EC.NONE			no correlation energy
EC.VWN5	LDA	[Vosko et al., 1980]	
EC.LDA	LDA	[Perdew and Wang, 1992]	
EC.LYP	GGA	[Lee et al., 1988]	
EC.PW91	GGA	[Perdew et al., 1992]	
EC.PBE	GGA	[Perdew et al., 1996]	
EC.HCTH93	GGA	[Hamprecht et al., 1998]	
EC.HCTH120	GGA	[Boese et al., 2000]	
EC.HCTH147	GGA	[Boese et al., 2000]	
EC.HCTH407	GGA	[Boese and Handy, 2001]	
EC.AM05	GGA	[Armiento and Mattsson, 2005]	
EC.PBESOL	GGA	[Perdew et al., 2008]	
EC.RGE2	GGA	[Ruzsinszky et al., 2009]	
EC.PBEINT	GGA	[Fabiano et al., 2010]	
EC.SG4	GGA	[Constantin et al., 2016]	
EC.MPBE	GGA	[Haas et al., 2010]	xc3 ( $\beta$ ) is required
EC.ACGGA	GGA	[Burke et al., 2014, Cancio et al., 2018]	
EC.VSXC	MGGA	[Van Voorhis and Scuseria, 1998]	
EC.PKZB	MGGA	[Perdew et al., 1999]	
EC.TPSS	MGGA	[Tao et al., 2003]	
EC.REVTPSS	MGGA	[Perdew et al., 2009]	
EC.MGGA.MS	MGGA	[Sun et al., 2013]	
EC.MVS	MGGA	[Sun et al., 2015a]	
EC.SCAN	MGGA	[Sun et al., 2015b]	
EC.SCANL	MGGA	[Mejia-Rodriguez and Trickey, 2018]	
EC.RSCAN	MGGA	[Bartók and Yates, 2019]	
EC.R2SCAN	MGGA	[Furness et al., 2020]	
EC.TM	MGGA	[Tao and Mo, 2016]	

Table 7.5: EC-switches

**line 2:** free format (only blanks are allowed as separator)

RPRINT, H-mod, FFTopt, LUSE

RPRINT	NR2V	no additional output
	R2V	Exchange-correlation ( <b>case.r2v</b> ), Coulomb ( <b>case.vcoul</b> ) and total potentials ( <b>case.vtotal</b> ) are written as $(r^2V)$ to a file for plotting with <b>lapw5/3ddens</b> (and the corresponding switch); use "VAL" for normalization in <b>case.in5</b> )
H-mod	EECE	Onsite Hartree-Fock (inside spheres) for selected electrons (see 4.5.8)
	HYBR	Onsite Hybrid functionals (inside spheres) (see 4.5.8)
FFTopt	IFFT	optional keyword, which lets you define the IFFT <sub>x</sub> mesh and an enhancement factor in the next line (necessary for <b>runeece.lapw</b> )
LUSE		optional l-max value for the angular grid used in 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 <b>runeece.lapw</b> )

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

IFFT<sub>x</sub>, IFFT<sub>y</sub>, IFFT<sub>z</sub>, IFFTfactor, iprint, NCON ncon

IFFT <sub>x,y,z</sub>		FFT-mesh parameters in x,y,z directions for the calculation of the XC-potential in the interstitial region. Usually set automatically in <b>init.lapw</b> (dstart). The ratio of the 3 numbers should be indirect proportional to the lattice parameters. (-1 -1 -1 determines these numbers automatically and takes only IFFTfactor into account)
IFFTfactor		Multiplicative factor to the IFFT grid specified above. It needs to be enlarged for highly accurate GGA or meta-GGA calculations as well as for systems with H atoms with small spheres.

`iprint` optional print switch. `iprint=0` will greatly reduce `case.output0` (in particular for **lapw0.mpi**).

`NCON` 9 optional keyword (NCON) and value for NCON (convergence parameter for pseudo charge density (between 4-18)) can be specified.



Keyword for $V_X$	Type	Reference	Comments
VX_NONE			no exchange potential
VX.LDA	LDA	[Kohn and Sham, 1965]	
VX.SLDA	LDA	[Tran and Blaha, 2011]	screened LDA for hybrid ( <b>case.in0.grr</b> )
VX.B88	GGA	[Becke, 1988]	
VX.PW91	GGA	[Perdew et al., 1992]	
VX.EV93	GGA	[Engel and Vosko, 1993]	
VX.LB94	GGA	[van Leeuwen and Baerends, 1994]	
VX.PBE	GGA	[Perdew et al., 1996]	
VX.REVPBE	GGA	[Zhang and Yang, 1998]	
VX.RPBE	GGA	[Hammer et al., 1999]	
VX.HCTH93	GGA	[Hamprecht et al., 1998]	
VX.HCTH120	GGA	[Boese et al., 2000]	
VX.HCTH147	GGA	[Boese et al., 2000]	
VX.HCTH407	GGA	[Boese and Handy, 2001]	
VX.AM05	GGA	[Armiento and Mattsson, 2005]	
VX.WC	GGA	[Wu and Cohen, 2006]	
VX.PBE-ALPHA	GGA	[Madsen, 2007]	
VX.PBESOL	GGA	[Perdew et al., 2008]	
VX.SOGGA	GGA	[Zhao and Truhlar, 2008]	
VX.RGE2	GGA	[Ruzsinszky et al., 2009]	
VX.PBEINT	GGA	[Fabiano et al., 2010]	
VX.OPTPBE	GGA	[Klimeš et al., 2010]	
VX.OPTB88	GGA	[Klimeš et al., 2010]	
VX.OPTB86B	GGA	[Klimeš et al., 2011]	
VX.HTBS	GGA	[Haas et al., 2011]	
VX.AK13	GGA	[Armiento and Kümmel, 2013]	
VX.CAP	GGA	[Carmona-Espindola et al., 2015]	
VX.SG4	GGA	[Constantin et al., 2016]	
VX.MPBE	GGA	[Haas et al., 2010]	xc1 ( $\mu$ ) and xc2 ( $\kappa$ ) are required
VX.B3PW91	GGA	[Becke, 1993]	semilocal part of hybrid B3PW91
VX.B3LYP	GGA	[Stephens et al., 1994]	semilocal part of hybrid B3LYP
VX.SPBE	GGA	[Tran and Blaha, 2011]	screened PBE for hybrid ( <b>case.in0.grr</b> )
VX.SWC	GGA	[Tran and Blaha, 2011]	screened WC for hybrid ( <b>case.in0.grr</b> )
VX.SPBESOL	GGA	[Tran and Blaha, 2011]	screened PBESol for hybrid ( <b>case.in0.grr</b> )
VX.SB88	GGA	[Tran and Blaha, 2011]	screened B88 for hybrid ( <b>case.in0.grr</b> )
VX.SHJSPBE	GGA	[Henderson et al., 2008]	screened PBE for hybrid ( <b>case.in0.grr</b> ), $\omega = (2/3)\lambda$
VX.SHJSPBESOL	GGA	[Henderson et al., 2008]	screened PBESol for hybrid ( <b>case.in0.grr</b> ), $\omega = (2/3)\lambda$
VX.SHJSB88	GGA	[Weintraub et al., 2009]	screened B88 for hybrid ( <b>case.in0.grr</b> ), $\omega = (2/3)\lambda$
VX.SHJSB97X	GGA	[Henderson et al., 2008]	screened B97x for hybrid ( <b>case.in0.grr</b> ), $\omega = (2/3)\lambda$
VX.GLLBSC	GGA, $\epsilon_i$	[Kuisma et al., 2010]	
VX.BR89	MGGA	[Becke and Roussel, 1989]	
VX.MBJ	MGGA	[Tran and Blaha, 2009]	
VX.LMBJ	MGGA	[Rauch et al., 2020]	
VX.GBJ	MGGA	[Tran et al., 2015a]	xc1 ( $\gamma$ ), xc2 ( $p$ ) and xc3 (0=no UC, 1=UC) are required. $c$ is in <b>case.in0abp</b>
VX.SMBJ	MGGA/nonlocal	[Becke and Johnson, 2006]	
VX.SLATER	nonlocal	[Slater, 1951]	
VX.KLI	nonlocal	[Krieger et al., 1990]	
VX.RS			$r_{s,\sigma} = (3 / (8\pi\rho_\sigma))^{1/3}$ written in <b>case.r2v</b> for plotting
VX.S			$s_\sigma =  \nabla\rho_\sigma  / ((3\pi^2)^{1/3} (2\rho_\sigma)^{4/3})$ written in <b>case.r2v</b> for plotting
VX.LAPRHO			$\nabla^2\rho_\sigma$ written in <b>case.r2v</b> for plotting
VX.TAU			$\tau_\sigma$ written in <b>case.r2v</b> for plotting (post-PBE calculation only)
VX.TAUTF			$\tau_\sigma^{TF}$ written in <b>case.r2v</b> for plotting
VX.TAUW			$\tau_\sigma^W$ written in <b>case.r2v</b> for plotting
VX.TAU-TAUW			$\tau_\sigma - \tau_\sigma^W$ written in <b>case.r2v</b> for plotting (post-PBE calculation only)
VX.Z			$\tau_\sigma^W / \tau_\sigma$ written in <b>case.r2v</b> for plotting (post-PBE calculation only)
VX.ALPHA			$(\tau_\sigma - \tau_\sigma^W) / \tau_\sigma^{TF}$ written in <b>case.r2v</b> for plotting (post-PBE only)
VX.ELF		[Becke and Edgecombe, 1990]	ELF = $\frac{1}{1 + ((\tau_\sigma - \tau_\sigma^W) / \tau_\sigma^{TF})^2}$ written in <b>case.r2v</b> for plotting (post-PBE only)
VX.GRR		[Tran and Blaha, 2009]	average of $ \nabla\rho  / \rho$ for mBJ ( <b>case.in0.grr</b> )
VX.LGRR		[Rauch et al., 2020]	local average of $ \nabla\rho  / \rho$ for lmBJ ( <b>case.in0.grr</b> )

Table 7.6: VX-switches

Keyword for $V_C$	Type	Reference	Comments
VC.NONE			no correlation potential
VC.LDA	LDA	[Perdew and Wang, 1992]	
VC.VWN5	LDA	[Vosko et al., 1980]	
VC.LYP	GGA	[Lee et al., 1988]	
VC.PW91	GGA	[Perdew et al., 1992]	
VC.PBE	GGA	[Perdew et al., 1996]	
VC.HCTH93	GGA	[Hamprecht et al., 1998]	
VC.HCTH120	GGA	[Boese et al., 2000]	
VC.HCTH147	GGA	[Boese et al., 2000]	
VC.HCTH407	GGA	[Boese and Handy, 2001]	
VC.AM05	GGA	[Armiento and Mattsson, 2005]	
VC.PBESOL	GGA	[Perdew et al., 2008]	
VC.RGE2	GGA	[Ruzsinszky et al., 2009]	
VC.PBEINT	GGA	[Fabiano et al., 2010]	
VC.SG4	GGA	[Constantin et al., 2016]	
VC.MPBE	GGA	[Haas et al., 2010]	xc3 ( $\beta$ ) is required
VC.ACGGA	GGA	[Burke et al., 2014, Cancio et al., 2018]	

Table 7.7: VC-switches

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

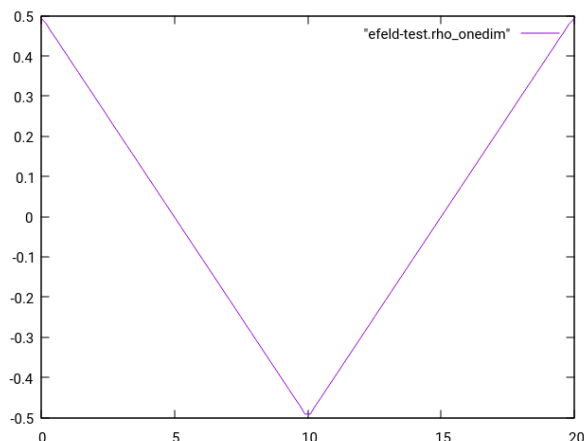
**line 4:** free format

IFIELD, EFIELD, WFIELD

**IFIELD** number of Fourier coefficients to model the zig-zag potential, also known as the ‘analytic triangular ramp’ (mode=0, default). Typically use IFIELD=30 (the maximal value is IFIELD=999, but you should stay way below that). An EFIELD of 1.0 for a lattice constant  $c = 20$  bohr gives you a field as shown in Fig. 7.1. Thus you should put the atoms of your slab centered around  $z=0.25$  (in the linear region of the zig-zag field) and the kinks should be in the vacuum. Make sure that you do not have inversion symmetry. Using IFIELD=-999 (with a random value for EFIELD) lists in case.output0 the other available (experimental) modes of electric fields. You can select mode  $n$  by specifying IFIELD= $n*1000$  (e.g. IFIELD=4000 gives you mode 4). The mode that is being used, is printed at the top of case.output0.

**EFIELD** value (amplitude) of the electric field. The electric field value EFIELD corresponds to an electric field in Volt/Angstrom, calculated as:  $EF[V/\text{Ang}] = 2*EFIELD/c * 13.6/0.529177$ , where  $c$  is your  $c$  lattice parameter ( $EFIELD = 1/2*EF[V/\text{Ang}]*c * 0.529177/13.6$ ).

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

Figure 7.1: Form of triangular electric field of 1 Ry for  $c = 20$  bohr

Alternatively, the 4th and following lines can be used for a staggered field (see section 4.5.4)

**line 4:** fixed format (starts at 1st column)

STAGFIELD (the presence of this keyword turns on a staggered field, making quasi FSM calculations for antiferromagnets possible)

**line 5:** free format

natom

natom    number of atoms for which a shift is applied

**line 6:** free format

iatom(i),shift(i)

iatom    index of atom in struct file

shift    value of the shift (in Ry) added to the spin-up potential (–shift is added to the spin-down potential)

**6th line repeated natom-times**

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

**dftd3** calculates the dispersion energy and forces using the DFT-D3 method of [Grimme et al., 2010, Grimme et al., 2011]. 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 <https://www.chemiebn.uni-bonn.de/pctc/mulliken-center/software>. When compilation is done, the executable **dftd3** has to be copied in the **\$WIENROOT** directory.

### 7.2.1 Execution

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

```
x dftd3
```

### 7.2.2 Input

The options for **dftd3** 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      95
cnthr       40
num         no
----- bottom of file: -----
```

A short summary of the options is given below and more details can be found in the manual of **dftd3**. Note that **case.indftd3** is read by the c-shell script **x.lapw** and that all data should be written in small letters.

- ▶ **method** : choice of the DFT-D method: **bj** (the recommended one), **zero**, **bjm**, **zerom** or **old** (which is the older DFT-D2 method).
- ▶ **func <functional>** : three choices are possible:
  - **default**, which means the functional specified in **case.in0**. Currently, this works for the following combinations in **case.in0**: “EX\_B88 EC\_LYP”, “XC\_PBE” (or “EX\_PBE EC\_PBE”), “EX\_REVPBE EC\_PBE”, “EX\_RPBE EC\_PBE”, “XC\_PBESOL” (or “EX\_PBESOL EC\_PBESOL”), “EX\_B88 EC\_PBE” and “EX\_TPSS EC\_TPSS”.
  - one of the functionals listed in the FORTRAN file **dftd3.f** (e.g., b-lyp or pbe)
  - **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 directory, or in **.dftd3par.local** in his working directory. For instance, to use the meta-GGA SCAN with DFT-D3, you should put the following parameters ([Brandenburg et al., 2016] into **.dftd3par.local**:
 

```
1.0 0.538 0.0 5.42 0.0 4
```
- ▶ **grad**: **yes** or **no** for the calculation of the forces on the nuclei (necessary for the minimization of internal parameters).

- ▶ **pbs** : **yes** or **no** for periodic boundary conditions (pbs). 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 DFT-D4 (Calculate the dispersion energy with DFT-D4)

**dftd4** calculates the dispersion energy and forces using the DFT-D4 method of [Caldeweyher et al., 2017, Caldeweyher et al., 2019, Caldeweyher et al., 2020]. 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 **dftd4** package is not included by default in **WIEN2k**, but can be downloaded from the website of the group of S. Grimme <https://www.chemiebn.uni-bonn.de/pctc/mulliken-center/software>. One can download a precompiled binary or recompile yourself, but the executable **dftd4** has to be copied in the **\$WIENROOT** directory.

### 7.3.1 Execution

The program **dftd4** is executed by invoking the command:

```
x dftd4
```

### 7.3.2 Input

The options for **dftd4** have to be specified in the input file **case.indftd4**. 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.indftd4 -----
func      pbe
grad      yes
pbs       yes
param     default
mbdscale  1.0
property  no
----- bottom of file: -----
```

A short summary of the options is given below and more details can be found in the manual of **dftd4**. Note that **case.indftd4** is read by the c-shell script **x\_lapw** and that all data should be written in small letters.

- ▶ **func** <functional> : one of the functionals listed in the file **dftd4-3.5.0/share/dftd4/parameters.toml** (e.g., **r2scan** or **pbe**).
- ▶ **grad** : **yes** or **no** for the calculation of the forces on the nuclei (necessary for the minimization of internal parameters).
- ▶ **pbs** : **yes** or **no** for periodic boundary conditions (pbs). It should be **no** for an isolated atom or molecule in a big box.
- ▶ **param** : instead of the default ones, four values for the damping parameters **S6**, **S8**, **A1** and **A2** can be specified.
- ▶ **mbdscale** : multiplication factor of the (atm or mbd) many-body dispersion energy.
- ▶ **property** : **yes** prints additional output into **case.scfdftd4**.

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

`n1vdw` 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]. See also the description in Sec.4.5.16. Note that `n1vdw` is also used to calculate the local average of  $\nabla\rho/\rho$ , which is used by the `lmBJ` potential (see Sec. 4.5.12 for details).

### 7.4.1 Execution

The program `n1vdw` is executed by invoking the command:

```
x n1vdw [-p -lmbj]
```

### 7.4.2 Input

The options for the `n1vdw` package have to be specified in the input file `case.inn1vdw`. A template can be found in `$WIENROOT/SRC.templates/`:

```
----- top of file: case.inn1vdw -----
1          kernel type
-0.8491    parameters of the kernel
25         plane-wave expansion cutoff GMAX
0.3        density cutoff rhoc
T          calculation of the potential (T or F)
10         plane-wave expansion cutoff GMAXpot for the potential
3.78      smearing parameter sigma (in bohr)
----- bottom of file: -----
```

The options are explained below:

- ▶ **kernel type** : 1 (for the analytical form of [Dion et al., 2004]), 2 (for the analytical form of [Vydrov and Van Voorhis, 2010, Sabatini et al., 2013]) or 3 (for the analytical form of [Terentjev et al., 2018]).
- ▶ **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]). 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]). Four values ( $b$ ,  $C_0$ ,  $C_1$  and  $C_2$ ) for kernel type 3 (e.g.,  $b = 10$ ,  $C_0 = 0.0093$ ,  $C_1 = 0.5$  and  $C_2 = 300$  for `PBEsol+rVV10s` [Terentjev et al., 2018]).
- ▶ **plane-wave expansion cutoff GMAX** :  $25 \text{ bohr}^{-1}$  is a relatively good value.
- ▶ **density cutoff rhoc** :  $0.3 \text{ bohr}^{-3}$  is a relatively good value.
- ▶ **potential** : if set to T (true), the potential is calculated.
- ▶ **plane-wave expansion cutoff GMAXpot for the potential** :  $10 \text{ bohr}^{-1}$  is a relatively good trade-off between speed and accuracy. Using a smaller value is not recommended. You may check the accuracy by using a higher value (e.g.,  $12 \text{ bohr}^{-1}$ ).
- ▶ **smearing parameter  $\sigma$**  : used only for the `lmBJ` potential (see Sec. 4.5.12).  $\sigma$  determines the degree of localization for the average of  $\nabla\rho/\rho$ .  $3.78 \text{ bohr}$  is the default value.

## 7.5 ORB (Calculate orbital dependent potentials)

This program was contributed by:

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

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

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

- ▶ LDA+U. There are three variants of this method, two of them are discussed in [Novák et al., 2001]
  1. LDA+U(SIC) - introduced by [Anisimov et al., 1993], with an approximate correction for the self-interaction correction. This is probably best suited for strongly correlated systems. Sometimes it was suggested in literature to use for a full potential method and GGA (and not just LDA) an “effective”  $U_{eff} = U - J$ ; setting  $J = 0$ . However, later works showed that  $J$  can still be an important contribution to obtain results in agreement with experiment.
  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 polarization is not needed, it is sufficient to run real version of LAPW1, which then automatically puts the orbital field equal to zero. For systems without the center of inversion, when LAPW1 must be complex, an extra averaging of the LDA+U potential is necessary.

- ▶ Orbital polarization. The additional potential has the form [Brooks, 1985, Eriksson and Johansson, 1989]:

$$V_{OP} = c_{OP} \langle L_z \rangle l_z \quad (7.1)$$

where  $c_{OP}$  is the orbital polarization parameter,  $\langle L_z \rangle$  is projection of the orbital momentum on the magnetization direction and  $l_z$  is single electron orbital momentum component  $z$  parallel to  $\vec{M}$ .

- ▶ Exact exchange and Hybrid methods: see [Tran et al., 2006] and 4.5.8

- Interaction with the external magnetic field. In this case the additional potential has a simple form:

$$V_{B_{ext}} = \mu_B \vec{B}_{ext} (\vec{l} + 2\vec{s}). \quad (7.2)$$

The interaction with the electronic spin is taken into account by shifting the spin up and spin down exchange correlation potentials in LAPW0 by the energy  $+\mu_B B_{ext} - \mu_B B_{ext}$ , respectively. The interaction of  $B_{ext}$  with spin could be as well calculated using the 'Fixed spin moment' method. For an interaction with the orbital momentum it is necessary to specify the atoms and angular momentum numbers for which this interaction will be considered. Caution is needed when considering interaction of the orbital momentum with  $B_{ext}$  in metallic or metallic-like systems. For the analysis see the paper by [Hirst, 1997]

PS: Igor Mazin pointed out, that the total energy (:ENE) of a spin-orbit calculation in case of an interaction with the external field also includes the external term  $-\mathbf{L} * \mathbf{B}_{ext}$ , where  $\mathbf{L}$  is the calculated orbital moment. Thus when plotting the total energy as function of external field, one has to add a  $+\mathbf{L} * \mathbf{B}_{ext}$  term in order to calculate the total magnetic susceptibility from the slope of this curve.

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

For more information see also section 4.5.7.

### 7.5.1 Execution

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

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

### 7.5.2 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

### 7.5.3 Input

The required input files (for both, DFT+U or EECE) can be generated most conveniently using the script **init\_orb\_lapw**(see Sec.5.2.17).

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

#### Input for all potentials

**line 1:** free format

```
nmod,natorb,ipr
```

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



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

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

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

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

### 3rd line repeated natorb-times

#### Input for LDA+U (nmod=1)

**line 4:** free format

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

**line 5:** free format

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

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

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

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

#### Input for Orbital Polarization (nmod=2)

**line 4:** (free format)

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

**line 5:** (free format)

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

**this line is repeated natorb-times**

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

```
pop(li,i)      OP parameter in Ry
```

**line 7:** (free format)

```
xms(1), xms(2), xms(3)
```

direction of magnetization expressed in terms of lattice vectors

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

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

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

**line 4:** (free format)

```
 $B_{ext}$       external field in Tesla
```

**line 5:** (free format)

```
xms(1), xms(2), xms(3)
```

direction of magnetization expressed in terms of lattice vectors

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

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

## 7.6 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 [Sjöstedt 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 -it``**. In some cases it is preferable to use **min\_lapw -j ``run\_lapw -I -fc 1 -it1``**, which will recreate **case.storeHinv**, or do not store these files at all using **min\_lapw -j ``run\_lapw -I -fc 1 -it -noHinv ``**

Parallel execution (fine grain and on the k-point level) is also possible and is described in detail in Sec. 5.5. A much faster alternative to ScalaPack is using 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.

### 7.6.1 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 `lapw1s.def` file uses a `case.in1s` file and creates the files `case.output1s` and `case.vectors`. For the spin-polarized case `lapw1` is called twice with `uplapw1.def` and `dnlapw1.def`. To all relevant files the keywords “up” or “dn” are appended (see the fcc Ni test case in the **WIEN2k** package).

## 7.6.2 Dimensioning parameters

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

LMAX	highest l+1 in basis function inside sphere (consistent with input in case.in1)
LMMX	number of LM terms in potential (should be at least NCOM-1)
LOMAX	highest l for local orbital basis (consistent with input in case.in1)
NMATMAX	maximum size of H,S-matrix (defines size of program, should be chosen according to the memory of your hardware, see chapter 11.2.3!)
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
NVEC2	K-vectors when multiplied with the reciprocal Bravais matrix
NVEC3	
restrict_output	for mpi-jobs, limits the number of case.output1_X_proc_XXX files to “restrict_output”

## 7.6.3 Input

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

```
----- top of file: case.in1 -----
WFFIL EF=0.5000 (WFPRI,WFFIL,SUPWF ; wave fct. print,file,suppress
 7.500 10 4 ELPA pxq BL 64 (R-mt*K-max; MAX l, max l for hns )
 0.30 5 0 (global energy parameter E(l), with 5 other choices, LAPW)
 0 -3.00 0.020 CONT 0 ENERGY PARAMETER for s, LAPW
 0 0.30 0.000 CONT 0 ENERGY PARAMETER for s-local orbital, LAPW-LO
 1 -1.90 0.020 CONT 0 ENERGY PARAMETER for p LAPW
 1 0.30 0.000 CONT 0 ENERGY PARAMETER for p-local orbitals LAPW-LO
 2 0.20 0.020 CONT 1 APW
 0.20 3 0 (global energy parameter E(l), with 1 other choice, LAPW)
 0 -0.90 0.020 STOP 0 LAPW
 0 0.30 0.000 CONT 0 LAPW-LO
 1 0.30 0.000 CONT 1 APW
K-VECTORS FROM UNIT:4 -9.0 2.0 69 emin/emax/nband
1.d-15 0.0 spro_limit for it.diag., lambda for it.diag
----- bottom of file -----
```

Interpretive comments follow:

**line 1:** free format  
switch, EF

- switch WFFIL standard option, writes wave functions to file `case.vector` (needed in `lapw2`)
- SUPWF suppresses wave function calculation (faster for testing eigenvalues only)

WFPRI prints eigenvectors to `case.output1` and writes `case.vector` (produces long outputs!)

EF optional input. If “EF=” key is present, lapw1 reads EF and sets 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, lnsmax, library, gridshape, bl, bd, memory

rkmax  $R_{mt} * K_{max}$  determines matrix size (convergence), where Kmax is the plane wave cut-off, Rmt is the smallest of all atomic sphere radii. Usually this value should be between 5 and 9 (APW+lo) or 6 - 10. (LAPW-basis) ( $K_{max}^2$  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  $R_{mt}$  (e.g. 0.7 a.u.), RKmax = 3 might already be reasonable, but convergence must be checked for a new type of system. For a hint of a reasonable  $R_{mt} * K_{max}$  identify which atom has the **smallest**  $R_{mt}$  and checkout the Table given at [http://www.wien2k.at/reg\\_user/faq/rkmax.html](http://www.wien2k.at/reg_user/faq/rkmax.html).

Note, that the actual matrix size is written on case.scf1. It is determined by whatever is smaller, the plane wave cut-off (specified with RKmax) or the maximum matrix dimension NMATMAX, (see previous section).

lmax maximum l value for partial waves used inside atomic spheres (should be between 8 and 12)

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

**NOTE:** the following parameters are optional and influence only the behavior/speed of mpi-parallel calculations

library ELPA ELPA is used to solve the eigenvalue problem in case of parallel calculations (default, several times faster than ScalaPACK).

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

Note, that this switch is only used, 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)

blocksize BL xx BLOCK size xx (default=192) for Hamilt and hns.F, may influence the cpu time in mpi-parallel calculations. BL=999 is an attempt to optimize this value automatically

blocksize BD xx BLOCK size xx (default=32) for diagonalization, may influence the cpu time in mpi-parallel calculations

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. (not active at the moment)

**line 3:** free format

Etrial, ndiff, Napw

Etrial	default energy used for all $E_l$ to obtain $u_l(r, E_l)$ as regular solution of the radial Schrödinger equation [used in equ.2.4,2.7] (see figure 7.2).
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)  
l, El, de, switch, NAPWL

l	l of partial wave
El	$E_l$ for $L=l$
de	energy increment de=0: this E(l) overwrites the default energy (from line 3) de $\neq$ 0: a search for a resonance energy using this increment is done. 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. 7.2. 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. Optionally one can change here the usage of APW to LAPW (change 1 to 0 after the CONT/STOP switch), since often APW is necessary only for orbitals which are more difficult to converge (3d, 4f) or have smaller $R_{mt}$ .

>>>**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. You can add multiple LOs at much higher energies (HELOs) for a better description of unoccupied states (see eg. for NMR calculations). In addition one can define ONE HDLO per l-value when the additional line has NAPWL=2. 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 ( $E_{LO} > -2.0Ry$ ) 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 (mainly for d- or f states) or HELOs (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.*

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

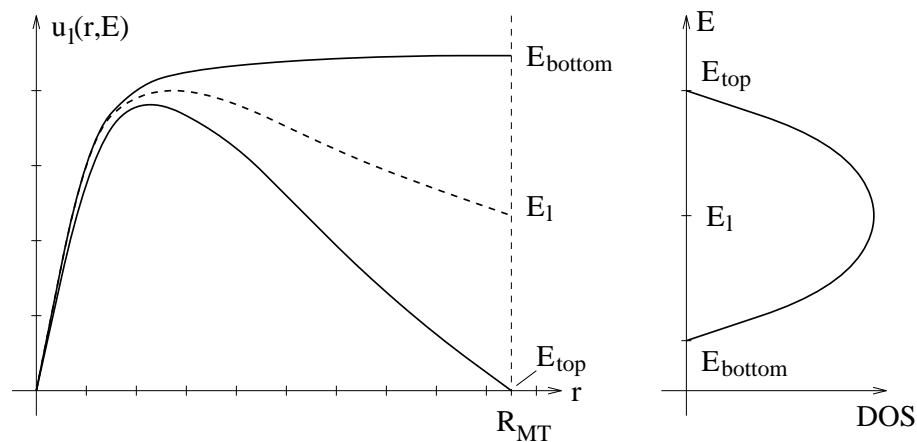


Figure 7.2: Schematic dependence of DOS and  $u_l(r, E_l)$  on the energy

**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 <b>case.klist</b> (generated by KGEN). Should not be changed.
EMIN, EMAX	energy window in which eigenvalues shall be searched (overrides setting in <b>case.klist</b> ). 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 <i>ispin</i> is 2 for non-spinpolarized and 1 for spinpolarized cases) in <b>lstart</b> and <b>init.lapw</b> . 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 $\lambda$ value for preconditioner of iterative diagonalization (see above). By default we use $\lambda = 0$ , but in some cases convergence can be improved by a small (around 1.0) positive or negative $\lambda$

**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 $2\pi/a$ , $2\pi/b$ , $2\pi/c$ for P, C, F and B cubic, tetragonal and orthorhombic lattices, but internal coordinates for H and monoclinic/triclinic lattices
weight	of k-vector (order of group of k)

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

>>>: **the last line** must be END

## 7.7 HF (Calculates the hybrid orbitals and eigenvalues)

**hf** calculates the orbitals and eigenvalues for hybrid functionals using the 2nd variational procedure, i.e., the semilocal orbitals generated by **lapw1** are used as basis functions for the 2nd 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**, **lmaxe** and **lmaxv**) 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 **lmaxe** and **lmaxv** 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. 4.5.9. Parallel execution (fine grain MPI and on the k-point level) is also possible and is described in Secs. 4.5.9 and 5.5. The use of **HDLOs** is not supported, eventually use **HELOs** instead, but be careful with the setting of the energy parameter.

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**).

### 7.7.1 Execution

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

```
x hf [-up/dn -c -p -band -diaghf -nonself -newklist -redklist
      -model/mode2 -so] or
hf hf.def or hfc hf.def
```

### 7.7.2 Input

```
----- top of file: case.inhf -----
0.25      alpha
T         screened (T) or unscreened (F)
0.165    lambda
xx       nband
```



```

6          gmax
3          lmaxe
3          lmaxv
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 unscreend HF requires a denser k-mesh than screened HF.

**line 3:** free format

$\lambda$                     screening parameter in bohr<sup>-1</sup>. This line should be present only if screening is set to T (true) in line 2. With the value  $\lambda = 0.165$  bohr<sup>-1</sup>, 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 sometimes 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 **-diaghf** 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 **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 represents 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 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.

## 7.8 LAPWSO (adds spin orbit coupling)

**lapwso** includes spin-orbit (SO) coupling in a 2nd variational procedure and computes eigenvalues and eigenvectors (stored in `case.vectorso`) using the scalar-relativistic wavefunctions from **lapw1**. 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 **lapw1** by increasing EMAX in `case.in1` (up to 10 Ry!) - if you are running MPI-parallel calculations with ELPA you have to increase `nband` in `case.in1` instead!. We also provide an additional basisfunction, namely a "relativistic-LO" (RLO) with a  $p_{1/2}$  radial wavefunction, which improves the basis and removes to a large degree the dependency of the results on EMAX and RMT (see [Kuneš et al., 2001]). They cannot be used together with HDLOs or HELOs, but are particular helpful for heavier atoms with semicore p-states. They must not be used for EFG calculations because they may add a diverging term at the nucleus which spoils the EFGs. 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, EECE or OP) can be added to the hamiltonian in a cheap and simple way.

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

- ▶ `case.struct`
- ▶ `case.klist`
- ▶ `case.kgen`
- ▶ `case.in2c`
- ▶ `case.in1`

We recommend to use `init_so` (see Sec.5.2.18) which helps you together with `symmetso` (see Sec.9.29) to setup spinorbit calculations.

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

### 7.8.1 Execution

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

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

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

## 7.8.2 Dimensioning parameters

The following parameters are used (collected in file **module.f**):

FLMAX	constant = 3
LMAX	highest l of wave function inside sphere (consistent with <b>lapw1</b> )
LOMAX	max l for local orbital basis
NRAD	number of radial mesh points

## 7.8.3 Input

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

```

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

```

Interpretive comments on this file are as follows:

**line 1:** format(A5)

switch

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

**line 2:** free format

LLMAX, IPR, KPOT

LLMAX                    maximum l for wavefunctions  
 IPR                      print switch, larger numbers give additional output.  
 KPOT    0                V(dn) potential is used for  $\langle dn|V|dn \rangle$  elements, V(up) for  
                            $\langle up|V|up \rangle$  and  $[V(dn)+V(up)]/2$  for  $\langle up|V|dn \rangle$ .  
                           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  $L=1$   
 de                     energy increment (see **lapw1**)  
 switch                used only if de.ne.0  
                       CONT calculation continues, even if either  $E_{top}$  or  $E_{bottom}$  are not found  
                       STOP calculation stops if not both  $E_{top}$  and  $E_{bottom}$  are found (especially use-  
                           ful for semi-core states)

>>>: **line 6** must be repeated “nlr” times (or should be omitted if nlr=0).

**line 7:** free format

noff, (iatoff(i),i=1,noff)

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

## 7.9 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 and valence stress contributions (time consuming) are calculated here. For systems without inversion symmetry you have to use the program **lapw2c** (in connection with **lapw1c**).

The partial charges for each state (energy eigenvalue) and each k-vector can be written to files **case.help031**, **case.help032** etc., where the last digit gives the atomic index of inequivalent atoms (switch **-help\_files**). Optionally these partial charges are also written to **case.qtl** (switch **-qtl**). 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 kinetic energy densities are written to **case.tauval**(switch **-tau**) 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.

### 7.9.1 Execution

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

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

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

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

## 7.9.2 Dimensioning parameters

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

IBLOCK	Blocking parameter (32-255) in <code>l2main.F</code> , optimize for best performance
LMAX2	highest <i>l</i> in wave function inside sphere (smaller than in <code>lapw1</code> , at present must be <code>.le. 10</code> )
LOMAX	max <i>l</i> for local orbital basis
NCOM	number of LM terms in density
NGAU	max. number of Gaunt numbers
NRAD	number of radial mesh points
<code>restrict_output</code>	for <code>mpi-jobs</code> , limits the number of <code>case.output2.X_proc_XXX</code> files to “ <code>restrict_output</code> ”

## 7.9.3 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, STR, STRF, 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
	FOR	same as TOT, but in addition a “Pulay” force contribution is calculated (this option costs some extra computing time and thus should be performed only at the final scf cycles, see <code>run_lapw</code> script in sec. 5.1.4)
	STR	same as TOT, but in addition the valence stress contributions are calculated. Very expensive, makes <code>lapw2</code> 10-100 slower, only when converged (see above).
	STRF	same as FOR, but in addition the valence stress contributions are calculated (see above).
	QTL	partial charges only (generates file <code>case.qt1</code> for DOS calculations), set automatically by switch <code>-qt1</code>
	EFG	computes decomposition of electric field gradient (EFG), contributions from inside spheres (the total EFG is computed in <code>lapw0</code> ), set automatically by switch <code>-efg</code> .

ALM	this generates two files, <b>case.radwf</b> and <b>case.almb1m</b> , where the radial wavefunctions and the $A_{lm}, B_{lm}, C_{lm}$ coefficients of the wavefunction inside spheres are listed. Do not set it manually, but using the <b>-alm</b> switch. The file <b>case.almb1m</b> can get very big.
ALMD	this generates two files, <b>case.radwf</b> and <b>case.almb1m</b> , where the radial wavefunctions and the $A_{lm}, B_{lm}, C_{lm}$ coefficients of the wavefunction inside spheres are listed in a format usable for the TRIQS DMFT code. Do not set it manually, but using the <b>-almd</b> switch. The file <b>case.almb1m</b> can get very big.
CLM	CLM charge density coefficients only
FERMI	Fermi energy only, this produces weight files for parallel execution and for the <b>optic</b> and <b>lapwdm</b> package, set automatically by switch <b>-fermi</b> .
>>>:	TOT and FOR are the standard options, QTL is used for density of states (or energy bandstructure) calculations, EFG for analysis, while FOURI, CLM are for testing only.
EECE	if set to "EECE", calculates the density for specified atoms and angular momentum only. Used for exact-exchange or hybrid-calculations, usually set automatically by <b>runsp.lapw -eece</b>

**line 2:** free format

```
emin, ne, esepermin, eseper0, iqtsave
```

emin	lower energy cut-off for defining the range of occupied states, can be set temporarily to "X" by switch <b>-emin X</b> or <b>-all X Y</b>
ne	number of electrons (per unit cell) in that energy range
esepermin	LAPW2 tries to find the "mean" energies for each $l$ channel, for both the valence and the semicore states. To define "valence" and "semicore" it starts at (EF - "esepermin") and searches for a "gap" with a width of at least "eseper0" and defines this as separation energy of valence and semicore
eseper0	minimum gap width (see above). The values esepermin and eseper0 will only influence results if the option <b>-in1new</b> is used
iqtsave	optional value, checks if the low-energy bandranges (below -2 Ry) are "narrow" (below 0.2 Ry) and stops (iqtsave=1 = default) / does not stop (iqtsave=0). You may have to switch it off for extreme pressures, because then you may have large band width even for semi-core states. In addition, iqtsave=-1 switches off the "QTL-B.gt.15" stop, i.e. the calculations will continue even when a possible ghoststate has been detected. Use with great care, it should be at best a temporary fix and should not appear in the final scf cycles.

**line 3:** format(a5,f10.5)

```
efmod, eval
```

efmod	determines how $E_F$ is determined
ROOT	$E_F$ is calculated and k space integration is done by root sampling (this can be used for insulators, but for metals poor convergence is found)
TEMP	$E_F$ is calculated where each eigenvalue is temperature broadened using a Fermi function with a broadening parameter of eval Ry. The total energy is corrected corresponding to T=0K. (e.g. eval=0.002 Ry gives good total energy convergence, but has no "physical" justification)

TEMPS	$E_F$ is calculated where each eigenvalue is temperature broadened using a Fermi function with a broadening parameter of eval Ry. The total energy is corrected by $-TS$ corresponding to the temperature specified by eval (e.g. eval=0.002 Ry corresponds to about 40 C)
GAUSS	$E_F$ is calculated as above but a Gaussian smearing method is used with a width of eval Ry. (e.g. eval=0.002 gives good total energy convergence, but has no "physical" justification).
TETRA	$E_F$ is calculated and k space integration is done by the modified (if eval is .eq. 0) or standard (eval .ge. 100) tetrahedron-method [Blöchl et al., 1994]. This "standard" scheme is recommended for <b>optic</b> . In this case the file <b>case.kgen</b> , consistent with the k-mesh used in <b>lapw1</b> , must be provided (see Sec. 7.6). This is the recommended option although convergence may be slower than with Gauss- or temperature-smearing. TETRA may lead to problems (wrong number 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 densities in a specified energy interval, can be set temporarily by switch <b>-all X Y</b> .
eval	when efm0d 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 efm0d is set to ALL, eval specifies the upper limit of the energy window (in Ry; can be set temporarily by switch <b>-all X Y</b> ), if efm0d is set to TETRA, eval .ge. 100 specifies the use of the standard tetrahedron method instead of the modified one (see above). By default, TETRA will average over partially occupied degenerate states at $E_F$ with a degeneracy criterium $D = 1.d-6$ . You can modify this by setting eval equal to your desired D (or 100+D).

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

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

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

jatom\_rho, nl\_rho, l\_rho

jatom\_rho        index of atom for which a specific density should be calculated  
 nl\_rho            number of angular momentum l-values  
 l\_rho             angular momentum l-values for which a specific density should be calculated

>>>**line 3b:** must be repeated nat\_rho times

**line 4:** format (121(I3,I2))

L,M

LM values of lattice harmonics expansion (equ. 2.10), defined according to the point symmetry of the corresponding atom; generated in SYMMETRY, MUST be consistent with the local rotation matrix defined in **case.struct** (details can be found in [Kara and Kurki-Suonio, 1981]). CAUTION: additional LM terms which do not belong to the lattice harmonics will in general not vanish and thus such terms must be omitted. Automatic termination of the  $lm$  series occurs when a second 0,0 pair appears within the list. When you change the  $l, m$  list during an SCF calculation the Broyden-Mixing is restarted in MIXER.

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

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

Table 7.56: LM combinations of “Cubic groups” ( $3\parallel(111)$ ) direction, requires “positive atomic index” in case.struct. Terms that should be combined [Kara and Kurki-Suonio, 1981] must follow one another.

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

Table 7.57: LM combination and local coordinate system of “non-cubic groups” (requires “negative atomic index” in case.struct)

**line 5:** free format



**GMAX** max. G (magnitude of largest vector) in charge density Fourier expansion. For systems with short H bonds ( $RMT(H) \leq 0.55$  bohr,  $GMAX=25$ ) and for small spheres of C, N, or O with  $RMT \leq 1.1$  bohr ( $GMAX=16$ ) larger values could be necessary and are set automatically during initialization. Calculations using GGA (potential option 13 or 14 in `case.in0`) should also employ a larger **GMAX** value (e.g. 14), since the gradients are calculated numerically on a mesh determined by **GMAX**. When you change **GMAX** during an scf calculation the Broyden-Mixing is restarted in **mixer**.

**line 6:** A4

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

NOFILE always calculate new list of K-vectors

## 7.10 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.

### 7.10.1 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`.

### 7.10.2 Dimensioning parameters

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

<b>NCOM</b>	number of LM terms in density
<b>NRAD</b>	number of radial mesh points
<b>NSYM</b>	order of point group

## 7.11 LAPWDM (calculate density matrix)

This program was contributed by:



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

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

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

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

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

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

### 7.11.1 Execution

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

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

### 7.11.2 Dimensioning parameters

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

FLMAX	constant = 3
LMAX	highest l of wave function inside sphere (consistent with <code>lapw1</code> )
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

### 7.11.3 Input

The required input files (for both, DFT+U or EECE) can be most conveniently generated using the script `init_orb_lapw` (see Sec.5.2.17).

A sample input for `lapwdm` is given below.

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

Interpretive comments on this file are as follows:

**line 1:** free format

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

**line 2:** free format

natom number of atoms for which the density matrix is calculated

**line 3:** free format

iatom, nl, l

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

**line 3 is repeated natom times t**

**line 4:** free format, optional

RINDEX, LSINDEX, 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

## 7.12 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.

### 7.12.1 Execution

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

```
lcore lcore.def or x lcore [-up|-dn] [-tau|-vresp]
```

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

### 7.12.2 Dimensioning parameters

The following parameter is listed in file `param.inc`:

NRAD            number of radial mesh points

### 7.12.3 Input

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

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

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

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

Interpretive comments on this file are as follows:

**line 1:** free format

nrorb, shift, iprint

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

**line 2:** free format

n, kappa, occup

n	principle quantum number
kappa	relativistic quantum number (see Table 6.6)

occup                    occupation number (including spin), fractal occupations supported

>>>: **line 2** is repeated for each orbital (nrorb times; see line 1)

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

**line 3:** free format

0                        zero indicating end of job

## 7.13 MIXER (adding and mixing of charge densities)

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

1. straight mixing as originally proposed by Pratt (52) with a mixing factor  $Q$

$$\rho_{new}(r) = (1 - Q)\rho_{old}(r) + Q\rho_{output}(r)$$

2. a Multi-Secant mixing scheme contributed by L. Marks (see [Marks and Luke, 2008]), in which all the expansion coefficients of the density from several preceding iterations (usually 6-10) are utilized to calculate an optimal mixing fraction for each coefficient in each iteration. It is very robust and stable (works nicely also for magnetic systems with 3d or 4f states at EF, only for ill-conditioned single-atom calculations you can break it) and usually converges at least 30 % faster than the old BROYD scheme.
3. Two new variants on the Multi-Secant method including a rank-one update (see [Marks, 2013, Marks, 2021]) 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, the atomic forces and the stress 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 mode MSR1a (or MSECa) atomic positions will also be mixed and thus optimized. This scheme can be a factor or 2-3 faster than the traditional optimization using **min.lapw** and is highly recommended. It still uses **case.inM** to control convergence or fix some positions, similar as **min.lapw**.

There is also a constraint optimization, which allows to optimized atomic positions under certain constraints in order to find saddle points or reaction barriers. It needs an extra file **case.constraint**, which is described in **SRC.mixer/Docs**.

### 7.13.1 Execution

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

```
mixer mixer.def or x mixer [-orb -eece -dftd3 -dftd4 -tau -vresp]
```

The different switches create lines in `mixer.def` for `case.dmatup/dn` (-orb), `case.vorbup/dn` (-eece), `case.tausum/up/dn` (-tau) and `case.vrespsum/up/dn` (-vresp). The switches -dftd3 or -dftd4 direct mixer to add the corresponding total energy contributions. 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.

### 7.13.2 Dimensioning parameters

The following parameters are collected in file `modules.f`:

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

### 7.13.3 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.d0 YES (PRATT/MSEC1/3/MSR1/a bg charge (+1 for additional e), NORM
 0.2 MIXING GREED
1.0 1.0 Not used, retained for compatibility only
999 8 nbroyd nuse
# STIFF, STIFFER, FAST
----- bottom of file -----
```

Interpretive comments on this file are as follows:

**line 1:** (A5,\*)

switch, bgch, norm

switch	MSR1	Recommended. A Rank-One Multisecant that is slightly faster than MSEC3 in most cases. For MSR1a see later.
	MSR1a	Similar to MSR1, but in addition it optimizes the atomic positions simultaneously (see Sect. 5.3.2)
	MSEC3	Multi-Secant scheme (Marks and Luke 2008). This is equivalent to DIIS, with trust regions added.
	MSEC4	similar to MSEC3 (above), but mixes the higher LM values inside spheres by an adaptive PRATT scheme. This leads to a significant reduction of program size and file size (case.broyd*) for unit cells with many atoms and low symmetry (factor 10-50) with only slightly worse mixing performance.
	MSR2	similar to MSR1 (above), but mixes only the L=0 LM value
	PRATT	Pratt scheme with a fixed greed
	PRAT0	Pratt scheme with a greed restrained by previous improvement, similar to MSEC3

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)
norm	YES NO	Charge densities are normalized to sum of Z 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. In case of too large charge fluctuations (divergence and stop of the scf cycle), Q can be reduced but this can lead to stagnation. One should rarely reduce this below 0.05.
-------	--	--

**line 3 (optional):** (free format)

f\_pw, f\_clm

f\_pw Not used, retained for input compatibility.

f\_clm Not used, retained for input compatibility.

**line 4 (optional):** (free format)

nbroyd, nuse

nbroyd Not used, retained for input compatibility.

nuse For all Multisecant methods: Only nuse steps are used during modified broyden (this value has some influence on the optimal convergence. Usually 6-10 seems reasonable and 8 is the default). For MSR1a of large cells sometimes 16 is better.

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

trust

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

STIFFER

FAST For easy cases to accelerate (also MSR1a).

**Several other advances switches are listed in SRC.mixer/Docs.**

In addition, **mixer** reads optional control files like **.pratt** or **.msec** (mixing factor), 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 on further control files consult **SRC.mixer/Docs**.

Note: If convergence cannot be reached:

- ▶ Check your setup (struct file, inputs). Wrong structures may never converge.
- ▶ For complicated systems (many (more than 20) atoms, in particular metallic systems with many 3d or 4f electrons, surfaces, magnetic structures, but also for metaGGA or mBJ calculations) 40 iterations may not be enough. Keep going (use the -NI switch). It is possible that one needs 100-200 cycles.
- ▶ Select a reasonable convergence criterium. While small systems can be converged to -ec/-cc 0.000001, this will NOT be possible for the systems mentioned above.
- ▶ Usually it may NOT help to reduce the mixing greed for MSR1, but you may uncomment the "STIFFER" line.
- ▶ In VERY FEW cases (eg. metaGGA or mBJ calculations for free atoms or large surfaces) it might be necessary to switch for some time to PRATT with extremely small mixing factors (eg. 0.01-0.05). Monitor :DIS carefully and make intermediate **save\_lapw**. With this setting you will NEVER reach TRUE convergence, but it may help temporarily.



---

# 8 Programs for analysis, calculation of properties, and geometry optimization

---

## Contents

---

8.1	afmsim	174
8.2	AIM	174
8.3	BerryPI	178
8.4	BROADENING	178
8.5	DIPAN	180
8.6	ELAST	182
8.7	FILTVEC	185
8.8	FSGEN	188
8.9	IRelast	188
8.10	IRREP	190
8.11	JOINT	191
8.12	KRAM	193
8.13	LAPW3	195
8.14	LAPW5	195
8.15	3DDENS	199
8.16	LAPW7	201
8.17	MINI	204
8.18	MSTAR	206
8.19	NMR	207
8.20	OPTIC	208
8.21	OPTIMIZE	211
8.22	PES	212
8.23	QTL	213
8.24	RENDOS	216
8.25	SPAGHETTI	216
8.26	TELNES3	219
8.27	TETRA	226
8.28	XSPEC	229

---

## 8.1 Afmsim (Virtual tip)

This program simulates an atomic-force-microscop (AFM) image using the “Electric field approximation” or the “Mirror-charge approximation (Virtual tip)” [Chan et al., 2009] using the calculated Coulomb potential (there is no explicit tip involved and one needs just an ordinary relaxed surface calculation).

### 8.1.1 Execution

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

```
afmsim afmsim.def or x afmsim
```

It requires an input file **case.inafmsim**, which must contain the keywords “GRAD-V” or “MIRROR” to select plotting  $\nabla V$  (electric field) or the mirror charge approximation [Chan et al., 2009]. In addition file **case.xsf**, which must contain the Coulomb potential on a 3D-grid above the surface (prepared by **3ddens**) must be present, where the data along  $z$  should be on a fine mesh, since we calculate gradients along  $z$ . It creates a file **case\_afmsim.xsf**, which can be plotted by **xcrysdn**. For plotting select a plane normal to your surface (typically 1 Å above the surface) and vary the height and min/max values for best contrast.

## 8.2 AIM (atoms in molecules)

This program was contributed by:

⇒ Javier D. Fuhr and Jorge O. Sofo  
 Instituto Balseiro and Centro Atomico Bariloche  
 S. C. de Bariloche - Rio Negro, Argentina  
 email: fuhr@cab.cnea.gov.ar and sofo@psu.edu  
 The original code has been significantly sped up by L.Marks (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 the topology of the electron density according to Bader’s “Atoms in molecules” theory. For more information see [Bader,] and [Sofo and Fuhr, 2001] and [http://wien2k.at/reg\\_user/textbook/aim\\_sofa\\_notes.pdf](http://wien2k.at/reg_user/textbook/aim_sofa_notes.pdf).

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. The latter gives the “atomic charge” and its “charge state” (the difference between the nuclear charge and the atomic charge) at the bottom of **case.outputaim**.

Using the **-dn** switch you can integrate the spin-density in the atomic basins to get uniquely defined magnetic moments of individual atoms, which do not depend on RMT (see [Tran et al., 2020]).

It is important that you provide a “good” charge density, i.e. one which is well converged with respect to LM MAX in the CLM-expansion (you may have to increase the default LM-list to LM=8 or 10) and with as little “core-leakage” as possible (see **lstart**, sect. 6.4), otherwise discontinuities appear at the sphere boundary.

### 8.2.1 Execution

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

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

### 8.2.2 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

### 8.2.3 Input

The input file contains "SWITCHES", followed by the necessary parameters until an END-switch has been reached.

Examples for "critical-point" searches and "charge-integration" are given below:

```
----- top of file: case.inaim -----
CRIT
1           # index of the atom (counting multiplicity)
ALL        # TWO/THRE/ALL /FOUR
3 3 3     # x,y,z nshells (of unit cells)
END
----- bottom of file -----
```

Interpretive comments on this file are as follows:

**line 1: A4**

CRIT	Keyword to calculate critical points
------	--------------------------------------

**line 2: free format**

iatom	index of the atom from where the search should be started. This count includes the multiplicity, i.e. if the first atom has MULT=2, the "second atom" has iatom=3 (Do not use simply the atom-numbers from <code>case.struct</code> )
-------	---

**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/Å<sup>3</sup>, ... (instead of bohr).

```
----- top of file: case.inaim -----
SURF
3 atom in center of surface (including MULT)
40 0.0 3.1415926536 theta, 40 points, from zero to pi
40 -0.7853981634 2.3561944902 phi
0.07 0.8 2 step along gradient line, rmin, check
1.65 0.1 initial R for search, step (a.u)
3 3 3 nshell
IRHO "INTEGRATE" rho
WEIT WEIT (surface weights from case.surf), NOWEIT
30 30 radial points outside min(RMIN,RMT)
END
----- bottom of file -----
```

Interpretive comments on this file are as follows:

line 1: A4

SURF Keyword to calculate the Bader surface.

line 2: free format

iatom index of the atom from where the search should be started. This count includes the multiplicity, i.e. if the first atom has MULT=2, the "second atom" has iatom=3 (Do not use simply the atom-numbers from **case.struct**)

line 3: free format

ntheta, thmin, thmax

ntheta number of theta directions for the surface determination. This (and nphi) determines the accuracy (and computing time).  
 thmin starting angle for theta  
 thmax ending angle for theta. If you have higher symmetry, you can change the angles thmin=0, thmax=π and use only the "irreducible" part, i.e. when you have a mirror plane normal to z (see case.outputs), restrict thmax to π/2.

**line 4:** free format

nphi, phimin, phimax

nphi     number of phi directions for the surface determination  
 phimin   starting angle  
 phimax   ending angle. (see comments for theta to reduce phi from the full  $0 - 2\pi$  integration).

**line 5:** free format

h0, frmin, nstep

h0        step in real space to follow the gradient ( $\sim 0.1$ )  
 frmin     defines the radius, for which the routine assumes that the search path has entered an atom, given as "rmin = frmin \* rmt" (0.8-1.0)  
 nstep     number of steps between testing the position being inside or outside of the surface (2-8).

**line 6:** free format

r0, dr0

r0        initial radius for the search of the surface radius (1.5)  
 dr0       step for the search of the surface radius(0.1)

**line 7:** free format

nxsh, nysh, nzsh

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

**line 8:** A4

IRHO       integrate function on "unit 9" (usually **case.clmsum**) inside previously defined surface (stored in **case.surf**).

**line 9:** A4

WEIT       specifies the use of weights in **case.surf**.

**line 9:** free format

npt        specifies number of points for radial integration outside the MT (30)

**line 8:** A4

END        specifies end of job.

### 8.3 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/rubel175/BerryPI>  
 email: rubelo@mcmaster.ca

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 spontaneous polarization, Born effective charges or piezoelectric constants using the Berry phase approach. It can also be used to characterize topological materials (Weyl semimetals). More details about its usage are given in Chapter 5.8.

It consists of a set of Python scripts (requires the NumPi library) and uses **wien2wannier** for the calculation of overlap integrals. The main steps of a `berrypi` call include:

- ▶ `x kgen -fbz`  
generate a k-mesh in the full Brillouin zone, default: 4 4 4
- ▶ `x lapw1 (-up/dn -orb -p)`  
Calculate wavefunctions for the new k-list
- ▶ `x lapwso (-up -orb -p)` (only when spin-orbit is included)
- ▶ `x lapw2 -fermi (-up/dn -p -so)` to find occupied bands
- ▶ `write_inwf`  
Prepare the input for w2w with the occupied band range
- ▶ `write_win -band nofile` case  
Create the input file for w2w
- ▶ `win2nnkp.py` case  
Generate the nearest neighbor list of k-points
- ▶ `x w2w (-up/dn -so -p)`  
Calculate the overlap matrix  $S_{mn}(k_j, k_{j+1})$
- ▶ `x w2waddsp` (only in case of spin-polarization and SO)
- ▶ `mmn2pathphase.py` case `x`  
Calculate the Berry phase along x-axis

For magnetic calculations the commands `write_inwf`, `write_win`, `win2nnkp.py` and `w2w` are performed separately for `-up/-dn`, except when `-sp.c` is specified when they are only performed for the “up” spin. A number of command line options exist, which can be viewed by using the command `berrypi -h`. More details can be found in Chapter 5.8

### 8.4 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**) or the **pes** program (broadening of **case.pes1** 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.broadspeci** or **case.pes1b**.

### 8.4.1 Execution

#### Execution

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

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

### 8.4.2 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 2.1500000000000000
0.6
dummy
0.0
0.0
0.0
```

line	value	explanation
1	GaN ...	Title (of no consequence for the calculation)
2	ELNES/ABS/EMIS/PES	Type of input spectrum
3	NC C1 C2	specification of input file: NC number of columns, C1 and C2 column to broaden
4	ESPLIT XINT1 XINT2	split energy (eV), XINT1—2 relative intensities of spectra (for overlaying L23 spectra in ELNES)
5	GA GB	Lorentian core hole lifetime of the two edges (eV)
6	W WSHIFT	W: type of valence broadening (0:none, 1:linear with $E/E_0$ , 2:Muller like $E^2$ , 3:Moreau); WSHIFT: edge offset (eV)
7	S	Gaussian spectrometer broadening FWHM in eV
8	dummy	dummy keyword for compatibility with <b>lorentz</b>
9-11	E0, E1, E2	quadratic energy dependent broadening (only used for type ELNES and EMIS when selecting valence broadening type W=2)

For details of broadening with  $W = 1,2,3$  see the references given in valencebroadening.f. For  $W=1$  we use  $\gamma = \text{energy}/E_0$  (typically  $E_0=10.0$ ). For  $W=2$ :  $\gamma E_1 * (\text{energy}/E_0)^2$ . Larger  $E_0$  and/or smaller  $E_1$  reduces broadening.

PES spectra are broadened only with Lorentzian (GA) and Gaussian (S) broadening.

## 8.5 DIPAN (Dipolar anisotropies)

This program was contributed by:

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

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

This program calculates the magnetic dipolar hyperfine field and the dipolar magnetocrystalline anisotropy by a direct lattice summation over the magnetic moments of all sites.

The magnetic field is given by

$$\vec{B} = \frac{\mu_0 \mu}{4\pi r^3} [3(\vec{n}\hat{r})\hat{r} - \vec{n}] \quad (8.1)$$

where  $\hat{r} = \vec{r}/r$ .

$\vec{n} = \vec{M}/M$  is direction of magnetization.

$\mu_0$  is permeability of free space;  $\mu_0 = 4\pi 10^{-7}$  H/m.

$\vec{B}$  is the dipolar field in T.

$\vec{\mu}$  is magnetic dipolar moment in  $\text{Am}^2 = \text{J/T}$ , assumed to be parallel to  $\vec{n}$ .

$r$  is in m.

We want to express  $\mu$  in Bohr magnetons  $\mu_B = 9.274078 \cdot 10^{-24}$  J/T and

$r$  in atomic units for length  $a_0$  (Bohr radius)  $a_0 = 5.2917706 \cdot 10^{-11}$  m.

Inserting in (1) gives

$$\vec{B} = 6.258463 \frac{\mu(\mu_B)}{r(\text{a.u.})^3} [3(\vec{n}\hat{r})\hat{r} - \vec{n}]. \quad (8.2)$$

Total dipolar field acting on atom  $i$  is given by the lattice sum

$$\vec{B}_i = 6.258463 \sum_j \frac{\mu_j}{r_j^3} [3(\vec{n}\hat{r}_j)\hat{r}_j - \vec{n}]. \quad (8.3)$$

Dipolar anisotropy energy is given by the sum

$$E_{an} = -\frac{1}{2V} \sum_j \vec{B}_j \vec{\mu}_j \quad (8.4)$$

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

$$E_{an}(\text{J/m}^3) = -\frac{3.129232 \cdot 10^7}{V(\text{a.u.})^3} \sum_j \vec{B}_j(\text{T}) \vec{\mu}_j(\mu_B) \quad (8.5)$$



### 8.5.1 Execution

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

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

### 8.5.2 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

### 8.5.3 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 <b>case.outputdipan</b> and <b>case.nn.dipan</b>

**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

## 8.6 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@univie.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].

### 8.6.1 Execution

The package is driven by three scripts:

- ▶ **init.elast:**  
 It prepares the whole calculation and should be run in a directory with a valid **case.struct** and **case.inst** file. It creates the necessary subdirectories **elast**, **elast/eos**, **elast/tetra**, **elast/rhomb**, **elast/result**, the templates for tetragonal and rhombohedral distortion and initializes the calculations using **init.lapw**.
- ▶ **elast.setup:**  
 It should be run in the **elast** directory, generates the distorted struct-files and **eos.job**, **rhomb.job** and **tetra.job**. These scripts must be adapted to your needs (spin-polarization, convergence,...) and run. **elast.setup** can be run several times (for different distortions,...).
- ▶ **ana.elast:**  
 Once all calculations are done, change into **elast/result** and run this script. The final results are stored in **elast/result/outputs**.
- ▶ **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**

## 8.6.2 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
10000      NUMBER_of_k-points
15          GMAX
NM          SPIN
NM          INST
-----
7.777777          0
7.655344          10
7.553434          20
----- bottom of file -----
```

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 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 **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          0.00001
spin        .FALSE.
parallel    .TRUE.
----- bottom of file -----
```

Interpretive comments on this file are as follows:

**line 1-3 (optional):** free formatdistortion,  $n$ 

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 ( <b>runsp_lapw</b> used instead of <b>run_lapw</b> in <b>eos.job</b> , <b>tetra.job</b> and <b>rhomb.job</b> )

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

parallel	.FALSE. default
	.TRUE. if <b>.machines</b> exists in the <b>elast/</b> directory then it will be copied into <b>pressure.x/eos</b> , <b>pressure.x/tetra</b> , <b>pressure.x/rhomb</b> directories

## 8.7 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**.

### 8.7.1 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.

### 8.7.2 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 <b>lapw1</b> )
LOMAX	maximum L value used for local orbitals (as in <b>lapw1</b> )

The parameter LMAX and LOMAX must be set precisely as in **lapw1**; all other parameters must not be chosen smaller than the corresponding parameters in **lapw1**.

### 8.7.3 Input file case.inf

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

## I. Global Selection Mode

```

- - - - - top of file case.inf - - - - -
3  1 17 33      # number of k-point list items, k-points
2  11 -18      # number of bands, band indices
- - - - - end of file - - - - -

```

Interpretive comments on this file are as follows.

- line 1:** free format  
 kmax ik(1) ... ik(kmax) Number of  $k$ -point list items, followed by the list items themselves. Positive list items mean selection of the  $k$ -point with the specified index; negative list items mean selection of a range of  $k$ -points with indices running from the previous list item to the absolute value of the current one. E.g. the sequence 2 -5 stands for 2, 3, 4, and 5.
- line 2:** free format  
 nmax ie(1) ... ie(nmax) Number of band index items, followed by the list items themselves. Again, positive list items mean selection of a single band index; negative list items mean selection of a range of band indices.

## II. Individual Selection Mode

```

- - - - - top of file case.inf - - - - -
2  :           # number of k-points
17  4  11 13 15 17 # k-point, number of bands, band indices
33  3  11 -14 18   # k-point, number of bands, band indices
- - - - - end of file - - - - -

```

Interpretive comments on this file are as follows.

- line 1:** free format  
 kmax the number of individual  $k$ -points to be selected. This number must be followed by any text, e.g. 'SELECTIONS' or simply ':', to indicate individual selection mode.
- line 2:** free format  
 ik nmax ie(1) ... ie(nmax) First the index of the selected  $k$ -point, then the number of band index items, followed by the list items for the current  $k$ -point themselves. Positive list items mean selection of the band with the specified index; negative list items mean selection of a range of band indices running from the previous list item to the absolute value of the current one. E.g. the sequence 3 -7 stands for 3, 4, 5, and 7.  
 This input line has to be repeated  $kmax$ -times.

## 8.8 FSGEN (Fermi-surface generation)

Unfortunately there is no really versatile tool for Fermi surface generation or analyzing FS properties. We have collected here a series of small programs together with some description on how to proceed to generate 2D-Fermisurfaces within WIEN.

- ▶ As usually, you have to run an scf cycle and determine a good Fermi-energy. "Good" means here a Fermi-energy coming from a calculation with a dense k-mesh.
- ▶ You should than create a mesh within a plane of the BZ, where you want to plot the FS. Some utility programs like **sc.fs.mesh**, (fcc, bcc, cxz.mon and hex are also available) may help you here, but only some planes of the BZ have been implemented so far. Please check these simple programs and modify them according to your needs. Copy the generated k-mesh **fort.2** to **case.klist**.
- ▶ Run **lapw1** with this k-mesh.
- ▶ Run **spaghetti** with input-options such that it prints the bands which intersect EF to **case.spaghetti.ene** (line 10, see sec. 8.25)
- ▶ Edit **case.spaghetti.ene** and insert a line at the top:  
 NX, NY, x-len, y-len, NXinter, NYinter, Invers, Flip  
 where  
 NX, NY are the number of points in the two directions  
 x-len, y-len are the length of the two directions of the plane (in bohr<sup>-1</sup>, you can find this in **case.spaghetti.ene**)  
 NXinter, NYinter: interpolated mesh, e.g. 2\*NX-1  
 Invers: 0/1: mirrors x,y  
 FLIP: 0/1: flips x,y to y,x
- ▶ Run **spagh2rho < case.spaghetti.ene** to convert from this format into a format which is compatible with the **case.rho** file used for charge density plotting. It generates files **fort.11**, **fort.12**, ... (for each band separately) and you should use your favorite plotting program to generate a contourplot of the FS (by using a contourlevel = 0). Alternatively you can use for plotting:
- ▶ Run **fsgen\_lapw 11 xx save\_filename**, which is a small shell script that can plot all fermi surfaces using the data-files **fort.11**, **fort.12**, ... **fort.xx** generated in the previous steps. It requires the public domain package **pgplot** and the contour-plot program **plotgenc**. (The latter can be obtained from [http://www.wien2k.at/reg\\_user/unsupported/](http://www.wien2k.at/reg_user/unsupported/), but you must have installed the **pgplot** library before.)

## 8.9 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., 2018]) calculates elastic constants for cubic, hexagonal, orthorhombic, tetragonal, monoclinic and rhombohedral symmetry using the second derivatives of poly-



nomial fits of the total energy versus applied strains. It replaces previous versions on our “un-supported software” page. In the latest version it can also calculate the elastic constants under pressure.

The package is driven by the following scripts:

- ▶ **set\_elast\_pressure** This is an optional step, to be done only when you want to calculate the elastic constants for non-zero pressure. Before you can do this, you need to have done a volume optimization, so that the equation of state (and thus the pressure-volume relation) is known. It checks for the necessary files **case.struct**, **case.outputeos** or **case.struct**, **data.pressure**. When **case.outputeos** is present, it asks the user interactively for which pressures he wants to calculate the elastic constants and generates **n pressure\_x/case** directories. This option should be used only for cubic cases, since it will create structures with constant  $c/a$ ,  $b/a$  ratio according to the previously calculated equation of states (Murnaghan fit). Alternatively, in **data.pressure** (make sure, there is no **case.outputeos**) you can specify a list of  $n$  “pressure,  $a$ ,  $b$ ,  $c$ ” (in GPa and bohr) values (which have been obtained previously from  $c/a$  and volume optimization) for which the calculations will be done. At the end, **set\_elast\_pressure** calls **set\_elast\_lapw** automatically.
 

The following steps must be done in the original **case** directory (for the elastic constants at the volume (pressure) of the original **case.struct** file or for all **n pressure\_x/case** directories.
- ▶ **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 of the structure defined in **case.struct** and 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 **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 **setupc** program as it calls the auxiliary programs **getcalljob** and **makestruct**.
- ▶ **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:
- ▶ **anaIR\_elast\_lapw:**

finds the symmetry of the original struct file and calls **ana\_elastc\_lapw** script and determines the **elastic constants**  $C_{ij}$  (using the auxiliary program **fitdivELC**) as well as the **Voigt**, **Reuss**, **Hill**, **Bulk**, **Shear** and **Young modulus**, the **Poisson ratio**, **Pugh’s ratio**, **Kleinmans’s**, ... **parameters**, and ... using the auxiliary program **Calparameter** (it determines the invers of elastic constants). Using data from the auxiliary programs **fitdivELC** and **MassRho**, **anaIR\_elast\_lapw** by using the auxiliary program **Calparameter** calculates the **Sound Velocity**, the **Debye temperature**, **Vickers hardness**, **Kleinmans’s parameter** and.... The main output file is **case.output\_elastic** which contains the elastic constants and elastic compliance constants (invers of the elastic constants matrix) as well as mechanical and thermodynamical properties. Finally, **stabilityJAC** will check elastic stability conditions and **ana\_elastorder\_lapw** will check the sensitivity of the results to the order of the polynomial fit (stored in file **output-order**). For monoclinic crystals, the **TWS** program trans-

forms the elastic constants from WIEN2k to STANDARD Cartesian coordinates (stored in file **STDEL<sub>C</sub>-matrix**).

After a first run you may check your results using more datapoints (more or different displacements). This can be done conveniently by **setupc**, 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 **setupc** 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 **anaIR.elast\_lapw** and check if your elastic constants are converged with respect to computational parameters.

This is only a brief introduction into **IRelast**. Additional information can be found in **\$WIENROOT/SRC\_IRelast/guide** and it is highly recommended to read and follow the corresponding documents. You can also find 3 videos on [http://www.wien2k.at/reg\\_user/unsupported/example-Mono-ZrO2-2022.mkv](http://www.wien2k.at/reg_user/unsupported/example-Mono-ZrO2-2022.mkv), [http://www.wien2k.at/reg\\_user/unsupported/example-MgO-under-pressure-5GPa-2022.mkv](http://www.wien2k.at/reg_user/unsupported/example-MgO-under-pressure-5GPa-2022.mkv) and [http://www.wien2k.at/reg\\_user/unsupported/bad-point-2022.mkv](http://www.wien2k.at/reg_user/unsupported/bad-point-2022.mkv).

## 8.10 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@fys.uio.no

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**.

### 8.10.1 Execution

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

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

## 8.10.2 Dimensioning parameters

The following parameters are listed 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

## 8.11 JOINT (Joint Density of States)

This program was contributed by:

⇒ Claudia Ambrosch-Draxl  
 Atomistic Modelling and Design of Materials  
 University Leoben  
 A-8700 Leoben, AUSTRIA  
 email: sol-office@physik.hu-berlin.de

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

This program carries out the BZ integration using the momentum matrix elements **case.symmat** calculated before by **optic**. The interband or the intraband contributions to the imaginary part of the dielectric tensor ( $\epsilon_2$ ) 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  $\epsilon_2$  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.

### 8.11.1 Execution

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

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

### 8.11.2 Dimensioning parameters

The following parameter is listed in files **param.inc**:

NSYM            order of point group  
MG0             number of columns (usually 9)

### 8.11.3 Input

An example is given below:

```
----- top of file: case.injoint -----
  1 9999 8            : LOWER,UPPER,upper-valence BANDINDEX
-0.0000 0.00100 2.0000 : EMIN DE EMAX FOR ENERGYGRID IN ryd
eV                    : output units eV / ryd
XMCD                 : omitt these 4 lines for non-XMCD
-49.88 -50.80        : core energies in Ry (grep :2P case.scfc)
  1.6 0.6            : core-hole broadening (eV) for both core states
  0.1                : spectrometer broadening (eV)
  4                  : SWITCH
  2                  : NUMBER OF COLUMNS
  0.1 0.1 0.3        : BROADENING (FOR DRUDE MODEL - switch 6,7)
----- bottom of file -----
```

Interpretive comments on this file are as follows:

**line 1: free format**

b1, b2,            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,             Energy window and increment in Ry (emin must not be negative)  
de,  
emax

**line 3: free format**

units    eV        output in units of eV  
         Ry        output in units of Ry

**line 4: optional line for XMCD, must be omitted for ``normal`` optic; free format**

XMCD              keyword for XMCD calculation, requires 3 more lines

**line 4xmcd: must be omitted for ``normal`` optic; free format**

E\_core1,           lower and higher core energies (in Ry, get them using eg. "grep :2P  
E\_core2            case.scf")

**line 4xmcd: must be omitted for ``normal`` optic; free format**

broad\_core1,      lifetime broadening (eV) of lower and higher core state  
broad\_core2

**line 4xmcd: must be omitted for ``normal`` optic;** free format

broad              spectrometer (Gaussian) broadening (eV)

**line 4+:** free format

switch	0	joint DOS for each band combination
	1	joint DOS as sum over all band combinations
	2	DOS for each band
	3	DOS as sum over all bands
	4	imaginary part of the dielectric tensor ( $\epsilon_2$ )
	5	imaginary part of the dielectric tensor for each band combination
	6	intraband contributions: number of "free" electrons per unit cell assuming bare electron mass (calculated around $E_F \pm 10 * de$ as defined in input line 4), plasma-frequency
	7	in addition to switch 6 the contributions from different bands to the plasma frequency are analyzed.

**line 5:** free format

ncol              number of columns

**line 6:** free format

broadening

x,y,z              broadening parameters (in units defined in line 3) for Drude-model

The band analysis for all options (switches 0, 2, 5, and 7) has been improved: For each tensor component additional files are created, where each column contains the contributions from a single band or band combination. The file names are e.g. `.Im_eps_xx.1`, `.Im_eps_xx.2`, or `.jdos.1` etc. where the number of files depend on the number of bands/band combinations.

*Warning: The number of band combinations might be quite large!*

## 8.12 KRAM (Kramers-Kronig transformation)

This program was contributed by:



Claudia Ambrosch-Draxl  
Atomistic Modelling and Design of Materials  
University Leoben  
A-8700 Leoben, AUSTRIA  
email: cad@unileoben.ac.at

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

The Kramers-Kronig analysis is carried out for the actual number of columns contained in the **case.joint[up|dn]** file. For each real component its imaginary counterpart is created and vice versa. All dielectric tensor components can be found in file **case.epsilon[up|dn]**. The real and

imaginary parts of the optical conductivity (in  $10^{15}/s$ ) are written to file `case.sigmak[up|dn]`. In addition, file `case. absorp` contains the real parts of the optical conductivity (in  $1/(\Omega cm)$ ) and the absorption coefficients. The loss function is also calculated (`case.e loss`), where for the previously calculated Plasma-frequencies 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  $\epsilon_1$ ) cannot be obtained by simply adding the spin-up and dn results to get the total contribution (see equations in Ambrosch 06). Thus, one should add up both spin contributions of  $\epsilon_2$  (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  $\sqrt{2}$ , 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`.

### 8.12.1 Execution

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

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

### 8.12.2 Dimensioning parameters

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

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

### 8.12.3 Input

An example is given below:

```
----- top of file: case.inkram -----
0.0  gamma for Lorentz broadening (in units selected in joint)
0.0  energy shift (scissors operator) (in units selected in joint)
1    add intraband contributions? yes/no: 1/0
12.60 plasma frequencies (for each ``column'' in case.injoint)
0.20  Gammas for Drude terms (for each ``column'' in case.injoint)
----- bottom of file -----
```

Interpretive comments on this file are as follows:

**line 1:** free format

EGAMM	Lorentz broadening (in energy units selected in joint)
-------	--

**line 2:** free format

ESHIFT	Energy shift (scissors operator) (in energy units selected in joint)
--------	--

**line 3:** free format

INTRA	0	Intraband contributions are not added
	1	Intraband contributions are added (requires plasma-frequencies calculated by <b>joint</b> using switch "6")

**line 4:** free format

EPL		Plasma-frequencies (calculated by <b>joint</b> using SWITCH=6 for all columns)
-----	--	--

**line 5:** free format

EDRU		Broadening for Drude terms (for all columns)
------	--	--

## 8.13 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. In spin-polarized cases you can calculate the structure factors for spin-up and dn and subtract them later for magnetic structure factors.

### 8.13.1 Execution

The program **lapw3** is executed by invoking the command:

```
lapw3 lapw3.def or lapw3c lapw3.def or x lapw3 [-tot/-val -up/-dn ]
```

### 8.13.2 Dimensioning parameters

The following parameters are listed 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

## 8.14 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- (switch **-none**) and total (**-add**) densities but also spin densities (difference up-dn, **-sub**). It is also possible to plot the total density (**case.clmsum**, **-tot**), the kinetic-energy density (**case.tausum**, **-tau**) or the Coulomb (**case.vcoul**, **-coulomb**), exchange-correlation (**case.r2v**, **-exchange**), vtau potential in scf mGGA (**case.r2v2**, **-exchange2**) and the total (**case.vtotal**, **-pot**) potential. 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 `xcrysdn`, you should use the program `3ddens` (see Sect. 8.15).

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. A probably much faster alternative is to use the program `3ddens` (see Sect. 8.15).

In order to understand the full workflow for a meaningful valence electron density (or difference density), it is strongly recommended that you use "Run Programs [□](#) Tasks [□](#) Electron density plots" from `w2web`, see the TiC example in Fig.3.7 .

### 8.14.1 Execution

The program `lapw5` is executed by invoking the command:

```
x lapw5 [-up|dn -diff -val|-tot -add|-sub|-none
-pot|-coulomb|-exchange|-exchange2/-halfr2v|-tau ] or
lapw5(c) lapw5.def
```

Some of these switches may change the `case.in5` file or put different files into `lapw5.def`.

### 8.14.2 Dimensioning parameters

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

<code>LMAX2</code>	highest L in in LM expansion of charge and potential
<code>NCOM</code>	number of LM terms in density
<code>NRAD</code>	number of radial mesh points
<code>NPT00</code>	number of radial mesh points beyond RMT
<code>NSYM</code>	order of point group

### 8.14.3 Input

An example is given below. You may want to use `XCRYSDEN` by T.Kokalj to generate this file (see sect. 9.31.1).

```
----- 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, A4 format
ANG VAL NODEBUG # ANG/ATU, VAL/TOT, DEBUG/NODEBUG
NONORTHO    # optional line: ORTHO|NONORTHO
```



```

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

```

Interpretive comments on this file are as follows:

**line 1:** free format

`ix,iy,iz,idv`      The plane and section of the plot is specified by three points in the unit cell, an origin of the plot, an x-end and an y-end. The first line specifies the coordinates of the origin, where  $x=ix/idv$ , ... in fractional units of the lattice vectors (except *fc*, *bc* and *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

`ix,iy,iz,idv`      coordinates of x-end

**line 3:** free format

`ix,iy,iz,idv`      coordinates of y-end (The two directions *x* and *y* must be orthogonal to each other unless *NONORTHO* is selected). Since it is quite difficult to specify those 3 points for a rhombohedral lattice, an auxiliary program **rhomb\_in5** is provided, which creates those points when you specify 3 atomic positions which will define your plane. The most convenient way to specify this plane (for a more complex structure) is using *XCryS-Den*, 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`

<code>switch</code>	<code>RHO</code>	charge (or potential) plots, no atomic density is used (regular case)
	<code>DIFF</code>	difference density plot (crystalline - superposed atomic densities), needs file <b>case.sigma</b> (which is generated with <i>LSTART</i> , see section 6.4)
	<code>OVER</code>	superposition of atomic densities, needs file <b>case.sigma</b>
<code>addsub</code>	<code>NO</code>	(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  $e/\text{\AA}^3$  (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**, changed automatically when “-tot” is used
- 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 line is optional): free format**

- PWONLY calculate the density also inside the spheres using the PW expansion (“pseudodensity”, don’t expect that this has any physical meaning)

**line 9 (this and the following lines are optional for the STM mode): free format**

MODE, level, axis

- MODE STM enables STM mode
- level the density level (typically  $10^4 - 10^5 e^-/\text{\AA}^3$ ). If this value is inappropriately 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 10: free format**

VIBR

- GAUSS Gaussian smearing
- DAMP damped Gaussian smearing
- LAPL
- DAMP alternate damped Gaussian smearing
- LAP2
- DAMP alternate damped Gaussian smearing
- BESS

**line 11:** free format (only for GAUSS or DAMP modes)

ANIS                    for  $\exp(-x*x/\text{Anis}(1))$  damping (in Å)

**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

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

## 8.15 3DDENS (3D electron density plots in whole cell)

This program generates the charge density (or the potential) in the whole conventional (or primitive) unit cell on a three dimensional grid which can be used for plotting with an external program that can read `.xsf`-files (e.g., XCrysden, VESTA). Depending on the input files one can generate valence (**case.c1mval**, switch **-val**) or total densities (**case.c1msum**, switch **-tot**) and **3ddens** supports the same switches as `lapw5` (see 8.14). Optionally one can simulate a constant current STM image, where a height profile (above a surface slab) corresponding to a specific electron density value can be extracted from the **case.xsf**-file and written to the **case.stm.xsf**-file.

### 8.15.1 Execution

The program **3ddens** is executed by invoking the command:

```
3ddens 3ddens.def        or        x 3ddens [-val|-tot -up|-dn|-none
-tau|-pot|-coulomb|-exchange|-exchange2]
```

### 8.15.2 Dimensioning parameters

The following parameters are listed in file **param.inc**:

LMAX2            highest L in in LM expansion of charge and potential  
 NCOM            number of LM terms in density  
 NRAD            number of radial mesh points

### 8.15.3 Input

An example is given below. You can find this template in **\$WIENROOT/SRC\_templates/case.in3d**.

```
----- top of file: case.in3d -----
100 100 100 # number of 3D-gridpoints in a-, b- and c-direction
0.0 0.0    # extend (pos.)/reduce (neg.) plotting grid by some fraction of the lattice vectors
0.0 0.0    # 2nd line: a-direction, 3rd line: b-direction, 4th line: c-direction
0.0 0.0    # all values set to 0.0: use full unit cell
add        # no: use file 9; add: sum of files 9 and 11; sub: difference of 9 and 11
conv       # calculate density in conventional (conv) or primitive (prim) cell
stm z 60 100 0.0001 # (optional) stmswitch, direction, start, end, density
----- bottom of file -----
```

Interpretive comments on this file are as follows:

**line 1:** free format

ix, iy, iz      The number of gridpoints along a-, b-, and c-direction used for the calculation (the direction is given by the lattice vectors of the used cell).

**line 2-4:** free format

ix, iy      Extension (positive values) or reduction (negative values) of the plotting grid - given is the percentage of the change with respect to the gridpoints of line 1  
 Line 2: Extension/reduction in -a and a direction.  
 Line 3: Extension/reduction in -b and b direction.  
 Line 4: Extension/reduction in -c and c direction.

**line 5:** free format

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 6:** free format

celltype    conv      calculates the electron density in the conventional unit cell (default).  
               prim      calculates the electron density in the primitive cell.

**line 7 (optional):** free format

stmswitch, direction, start, end, density

stmswitchstm      case\_stm.xsf file is written, containing height values for the given density value.  
 direction    x | y |      specifies the direction perpendicular to the considered surface.  
               z  
 start      specifies the first plane for the search of the given density (typically a value above the surface atom).  
 end      specifies the last plane for the search of the given density (typically around middle of vacuum).  
 density      density that corresponds to the desired constant current ( $\sim 0.0001$  to  $0.00001$ ).

## 8.16 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**.

### 8.16.1 Execution

The program **lapw7** is executed by invoking the command:

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

### 8.16.2 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.

### 8.16.3 Input

A sample input is given below. It shows how to plot a set of wave functions on a 2-dim. grid.

```

----- top of file -----
2D ORTHO      # mode      O(RTHOGONAL)|N(ON-ORTHOGONAL)
0 0 0 2      # x, y, z, divisor of origin
3 3 0 2      # x, y, z, divisor of x-end
0 0 3 2      # x, y, z, divisor of y-end
141 101 35 25 # grid points and echo increments
NO           # DEP(HASING)|NO (POST-PROCESSING)
RE ANG LARGE # switch  ANG|ATU|AU  LARGE|SMALL
1 0         # k-point, band index
----- 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. $n = 0, 1, 2,$ or $3$ .
flag	orthogonality checking flag (for $n$ -dim. grids only)
N	The axes of the $n$ -dim. grid are allowed to be non-orthogonal.
O or (blank)	The axes of the $n$ -dim. grid have to be mutual orthogonal.

**line 2:** free format — (for  $n$ -dim. grids only)

ix iy iz idiv

Coordinates of origin of the grid, where  $x=ix/idv$  etc. in units of the *conventional* lattice vectors.

**line 3:** free format — (for  $n$ -dim. grids with  $n > 0$  only)

ix iy iz idiv

Coordinates of the end points of each grid axis. This input line has to be repeated  $n$ -times.

**line 4:** free format — (not for 0-dim. grids)

np ... In case of an  $n$ -dim. grid, first the number of grid points along each axis, and then the increments for the output echo for each axis. Zero increments means that only the first and last point on each axis are taken. In case of an arbitrary list of grid points, the total number of grid points and the increment for the output echo. Again a zero increments means that only the first and last grid point are taken. Hence, for  $n$ -dim. grids, altogether,  $2 * n$  integers must be provided; for arbitrary lists of grid points two integers are expected.

**line 5:** format(A3)

tool post-processing of the wave functions  
 DEP Each wave function is multiplied by a complex phase factor to align it (as most as possible) along the real axis (the so-called DEP(hasing) option).  
 NO No post-processing is applied to the wave functions.

**line 6:** format(A3,1X,A3,1X,A5)

switch iunit whpsi

switch the type of wave function data to generate  
 RE The real part of the wave functions is evaluated.  
 IM The imaginary part of the wave functions is evaluated.  
 ABS The absolute value of the wave functions is evaluated.  
 ARG The argument the wave functions in the complex plane is evaluated.  
 PSI The complex wave functions are evaluated.  
 iunit the physical units for wave function output  
 ANG Å units are used for the wave functions.  
 AU or Atomic units are used for the wave functions.  
 ATU  
 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 or data is generated in this case. This option is only allowed if a *single* 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 func-  
 PLOT tion as the one who's augmentation coefficients are stored in **case.abc**  
 is selected in the previous input line.  
 anything Augmentation coefficients are generated from the wave function infor-  
 else mation in **case.vector(f)**.

## 8.17 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], 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 **.minhess**).

### 8.17.1 Execution

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

```
mini mini.def or x mini
```

### 8.17.2 Dimensioning parameters

The following dimensioning parameters are collected in the file **param.inc**:

MAXIT	maximum number of geometry steps
NRAD	number of radial mesh points
NCOM	number of LM terms in density
NNN	number of neighboring atoms for nn
NSYM	order of pointgroup



### 8.17.3 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 2.0 (PORT/NEWT tolf step0 (a4,2f5.2))
1.0 1.0 1.0 3.0 ( 1..3:delta, 4:BO/eta(1=friction zero))
1.0 1.0 1.0 6.0 ( 1..3=0 constraint)
----- bottom of file -----
```

Interpretive comments on this file are as follows.

#### line 1: format(a4,2f5.2)

MINMOD	Modus of the calculation
PORT	Geometry optimization with reverse-communication trust-region Quasi-Newton routine from the Port library. Recommended option.
NEW1	Performs geometry optimization with "sophisticated" steepest-descent method with automatic adaptation of stepsize (still experimental, but when PORT fails, an interesting alternative)
NEWT	Performs geometry optimization with damped Newton scheme according to $R_m^{\tau+1} = R_m^\tau + \eta_m(R_m^\tau - R_m^{\tau-1}) + \delta_m F_m^\tau$ where $R_m^\tau$ and $F_m^\tau$ 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.
TOLP	(active only in -MRS1a mode of mixer). Position tolerance, geometry optimization will stop when all atomic movements are below TOLP.

#### 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 <b>case.struct</b> and the forces <b>:FGL</b> from <b>case.scf</b> ). <i>Whenever you change these DELTA(i) you must remove file <b>case.tmpM</b>!</i>
ETA	For NEWT: damping (friction) parameter. ETA=1 means no friction, ETA=0 means no speed from previous time steps

PORT: changes the strength of the bonds when running pairhess and ZWEIGHT is negative (see the pairhess description), otherwise not used  
 NEW1: ETA is not used

>>> **line 2:** must be repeated for every atom

### Input for Molecular dynamics:

```
----- top of file: nbc.inM -----
NOSE                               (NOSE/MOLD (a4))
58.9332 400. 1273. 5.0             (Masse, delta t, T, nose-frequency)
58.9332 400. 1273. 5.0
58.9332 400. 1273. 5.0
58.9332 400. 1273. 5.0
58.9332 400. 1273. 5.0
58.9332 400. 1273. 5.0
----- bottom of file -----
```

Interpretive comments on this file are as follows.

**line 1:** format(a4,f5.2)

MINMOD	Modus of the calculation
MOLD	Performs next molecular dynamics timestep
NOSE	Performs next molecular dynamics timestep using a NOSE thermostat

**line 2:** free format

MASS	Atomic mass of $i^{th}$ atom
TIMESTEP	Time step of MD (in atomic units, depends on highest vibrational frequencies)
TEMP	Simulation Temperature (K)
NOSF	Nose-frequency

>>>**line 2:** must be repeated for every atom

## 8.18 MSTAR (effective masses)

This program was contributed by:

⇒ Oleg Rubel  
 rubelo@mcmaster.ca  
 see M.Rubel et al., Computer Physics Communications 261 (2021) 107648  
 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 effective masses using a perturbative **k.p** approach. The formalism and its usage is described in detail in [Rubel et al., 2021]. It uses the momentum matrix elements (**case.mommat2**) from program **optic** (use a large  $E_{max}$  and "ON" in **case.inop**) and generates 4 files (**minv.ij.dat**, **minv.pr.dat**, **minv.c.dat**, and **minv.d.dat**, see [Rubel et al., 2021] for details).

One can reformat **minv.c.dat** into a WIEN2k-"qtl"-format using **x mstarqtl**, which can then be used in **spaghetti** to indicate the masses in a band structure plot.


### 8.18.1 Execution

The program **mstar** can be called only after **optic** and is executed by invoking the command:

```
x mstar [-up/-dn -settol 1.d-5] or
mstar case.mommat2 1.d-5
```

## 8.19 NMR (chemical shielding)

This program was contributed by:



Robert Laskowski  
 Inst.Materials Chemistry  
 TU Vienna  
 A-1060 Vienna, AUSTRIA

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 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.

## 8.20 OPTIC (calculating optical properties)

This program was contributed by:

⇒ Claudia Ambrosch-Draxl  
 Atomistic Modelling and Design of Materials  
 University Leoben  
 A-8700 Leoben, AUSTRIA  
 email: cad@unileoben.ac.at

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

The theoretical background is described in detail in [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

$$M_i = \langle n' \vec{k} | \vec{p} \cdot \vec{e}_i | n \vec{k} \rangle^2$$

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** will 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:

- ▶ generate a k-mesh in the whole Brillouin zone (x **kgen -fbz**);

- ▶ change TOT to FERMI in **case.in2c**;
- ▶ set IPRINT=1 in **case.inc** to activate core-wavefunction output;
- ▶ for metallic systems, put **TETRA** with value 101;
- ▶ execute **x lapw1 -up / -dn**;
- ▶ execute **x lapwso -up**;
- ▶ execute **x lapw2 -fermi -so -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.

Note that **OPTICS** should not be used with the hybrid functionals (see Sec. 4.5.9), since the full expression for the momentum matrix elements is not implemented yet and the matrix elements are incomplete.

### 8.20.1 Execution

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

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

**optic** is parallelized with OpenMP and over k-points.

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]**
- ▶ execute **run[sp]\_lapw -so -s lapw1 -e lcore** with this fine k-mesh
- ▶ run optic: **x optic -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
- ▶ cp case.energyso case.energysoup
- ▶ x lapw2 -fermi -so -up
- ▶ x optic -so -up
- ▶ x joint -up

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.

### 8.20.2 Dimensioning parameters

The following dimensioning parameters (listed in `param.inc_r` and `param.inc_c`) are used:

LMAX	highest l+1 in basis function inside sphere (reducing LMAX to 4 or 5 may dramatically speed-up optics for large cases, but of course the matrix elements will be truncated and do not have full precision)
LOMAX	highest l for local orbital basis (consistent with input in case.in1)
NRAD	number of radial mesh points
NSYM	order of point group

### 8.20.3 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 calculated
emax	
nbvalmax	optional input. Setting this to the number of occupied bands (see case.output2) will reduce cpu-time of optics (for large cases, MM only between occupied and empty bands)

#### line 3: optional line, must be omitted for ``normal`` optic; free format

XMCD	fixed keyword to indicate XMCD calculation. You should also use NCOL=6
natom	atom number (from <code>case.struct</code> file) for which XMCD should be calculated
edge	specify the edge: must be K, L1, L23, M1, M23 or M45

#### line 3+: free format

ncol	number of choices (columns in case.symmat)
------	--

#### line 4+: free format

icol	column to select. Choices are:
------	--------------------------------

```

1 ... Re < x >< x >
2 ... Re < y >< y >
3 ... Re < z >< z >
4 ... Re < x >< y >
5 ... Re < x >< z >
6 ... Re < y >< z >
7 ... Im < x >< y >
8 ... Im < x >< z >
9 ... Im < y >< z >

```

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

**line 5:** free format

IMME, NATOMS (optional input)

IMME	OFF/ON; optionally prints unsquared momentum matrix elements to unit 4
NATOMS	either "PW" or 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 $\epsilon_2$ . I recommend to repeat this analysis for all possible combinations, and also for a list of "all" atoms, since this shows the effect of the interstitial (and crossterms involving the interstitial).

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

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

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

IFPMAT	If IMME = "ON" and IFPMAT = 1: print the momentum matrix elements to <b>case.pmat</b> .
--------	---

## 8.21 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).

### 8.21.1 Execution

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

```
x optimize [-job "run -ec 0.00001 ..." -save savename] or
optimize optimize.def
```

The optional `-job` or `-save` switches puts the corresponding commands/names into the `optimize.job` script.

### 8.21.2 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 ( $\pm xx\%$ ) should be for each case.

## 8.22 PES (Calculate valence-band photo-electron spectra)

This program was contributed by:

⇒ Mahdiyar Bagheri  
 Inst. for Materials Chemistry, TU Vienna, Austria  
 email: mahdiyar4@gmail.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 program calculates the valence-band photo-electron spectrum (PES) according to [Bagheri and Blaha, 2019]. The program `pes` takes the calculated partial Density of States and multiplies them with the corresponding atomic cross-sections [Yeh and Lindau, 1985, Trzhaskovskaya et al., 2001, Trzhaskovskaya et al., 2006] to generate the PES spectrum. It takes into account the multiplicity of the different atomic sites and allows to normalize the PES spectra according to the amount of the corresponding partial charge of the free atom inside its atomic spheres and thus takes the localization/delocalization of wave functions into account.

### 8.22.1 Execution

The program `pes` requires `case.struct`, `case.outputst`, `case.int` and the partial density of states `case.dos1ev`, `case.dos2ev`, ... as input.

You need to generate the partial DOS for ALL atoms and ALL relevant “chemical angular momenta” (eg. C-s and C-p; or Ti-s and Ti-d) using the `tetra` program. Sometimes it is necessary to include also orbitals which are not occupied in free atoms (like Ti-4p). The program will allow you to estimate cross sections for these states using the information of neighboring atoms. Sometimes you will need to change the orbitals interactively, since the “essential” atomic orbital might be different (like the high-lying semicore Na-2s,2p states and not Na-3s,3p, which are empty in ionic solids).

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

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

You are asked interactively for a couple of inputs:




- ▶ specify an X-ray excitation energy (since the atomic cross-sections are energy dependent and the results can differ significantly for different excitation energies). Experimental spectra are often measured using Al-K $_{\alpha}$  (1486.6 eV) or Mg-K $_{\alpha}$  (1253.6 eV) radiation when performed in a lab, otherwise synchrotron radiation of a certain energy can be used.
- ▶ optionally reconfigure the relevant atomic orbitals (like Na-2p instead of 3p as mentioned above). An editor is opened in this case.
- ▶ which cross section database to use (usually Trzhaskovskaya et al [Trzhaskovskaya et al., 2001, Trzhaskovskaya et al., 2006] is the recommended choice, except for UPS at very low energy.
- ▶ for the (default) Trzhaskovskaya database you can also specify a specific polarization (see [Bagheri and Blaha, 2019]).
- ▶ If no atomic cross section is available (eg. for Zn-4p), you can extrapolate it from neighboring atoms (Ga, Ge) and estimate it yourself. An editor is opened in this case.
- ▶ use (recommended) or do not use the renormalization of the partial DOS by the inverse of the fractional charge inside the atomic spheres.
- ▶ optionally optimize the fractional charges such that the sum of the partial DOS comes close to the total DOS (to account for the missing interstitial). In this case, also a renormalized DOS file (**case.dosrn1ev**) is created (and can be plotted **dosplot2.lapw -ren**).

The summary is written into **case.outputpes**, while the total PES spectrum as well as its partial contributions are in **case.pes1/2/...** (see header of these files). The format of these files is identical to the corresponding dos-files, such that they can be plotted using **dosplot2.lapw -pes**.

## 8.23 QTL (calculates special partial charges and population matrices)

This program was contributed by:


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

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

**qtl** creates the input for calculating total and projected density of states of selected atoms (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.qtltext** [**up/dn**] is created in which the ordering of partial charges is given. *Please note, that in contrast to **case.qtl** [**up/dn**] 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 be NOT correct.*

### 8.23.1 Execution

The program **qt1** is executed by invoking the command:

```
x qt1 [ -up/dn -so -p -hf] or
qt1 qt1.def
```

### 8.23.2 Input

A sample input (a default is created automatically during **init\_lapw** for **case.inq** is given below.

```
----- top of file: case.inq -----
-7. 2.          Emin Emax
2              number of selected atoms
1 2 0 0        iatom1  qsplit1  symmetrize loro
2 1 2          nL1 p d
3 3 1 1        iatom2  qsplit2  symmetrize loro
4 0 1 2 3      nL2 s p d f
1. 1. 1.       new axis z
----- bottom of file -----
```

Interpretive comments on this file are as follows:

<b>line 1:</b> free format emin,emax	energy window
<b>line 2:</b> free format natom	number of atoms selected for calculation (max. 28, if more are needed you have to run qt1 in "junks")
<b>line 3:</b> free format iatom, QSPLIT, symmetrize, loro iatom QSPLIT symmetrize loro	integer, index of atom integer, analog of ISPLIT in <b>case.struct</b> : see below integer, =0 (no symmetrization), 1 (symmetrization) integer =0 original coord. system preserved =1 (new z axis) =2 (new z and x axes)
<b>line 4:</b> free format Nl(iatom), (l(iatom,i),i=1,Nl(iatom)) Nl l	number of orbital numbers selected for calculation orbital numbers selected for calculation for atom iatom
<b>line 5:</b> 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

Table 8.101: Possible values of QSPLIT and their interpretation

QSPLIT	meaning
-2	DOS in basis according to ISPLIT from case.struct
-1	DOS in relativistic $ j, l, s, m_j\rangle$ basis
0	DOS in relativistic $ j, l, s, m_j\rangle$ basis, summed over $m_j$
1	DOS in $ l, m_l\rangle$ basis (no symmetry)
2	DOS in basis of real orbitals (no symmetry)
3	axial symmetry
4	hexagonal symmetry
5	cubic symmetry
6	user written unitary transformation
88	population matrix, no $\langle l l'\rangle$ crossterms corresponds to ISPLIT=88
99	full population matrix including $\langle l l'\rangle$ crossterms (as ISPLIT=99)

the lattice vectors, need not be normalized). If not only the z axis, but also the x axis need to be specified, then loro must be equal to 2 and additional line

hx, kx, lx (real\*8)

giving the direction of the new axis x, perpendicular to the new axis z must appear. For relativistic splitting (QSPLIT=0,-1) this rotation is ignored and z points along the direction of magnetization as defined in `case.inso`.

Indices of selected atoms, as well as the orbital numbers, must form an ascending sequence.

For QSPLIT=6 (unitary transformation prepared by user) the unitary matrices are read as in WIEN2k\_07 `qtl`: For the i-th atom selected for qtl calculation, they are stored in `case.cf$i` and ordered according to increasing  $l$ . The unitary transformation matrix must rotate from the standard  $l_{ms}$ -basis to the desired one. A few examples (e.g.  $j_{jz}, l_{ms}$ , or  $e_g - t_{2g}$ ) 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.

### 8.23.3 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:

Ordering of DOS in QTL file for: HoMnO3 (Munoz)

```

atom  1 ordering of projected DOS
p,px,py,pz, real basis
d,dz2,d(x2-y2),dxy,dxz,dyz, real basis

atom  3 ordering of projected DOS
s
p,pxy,pz, axial basis
d,dz2,d(x2-y2),d(yz+xz),dxy, axial basis
f,A2,[x(T1)+y(T1)],z(T1),[ksi(T2)+eta(T2)],zeta(T2), axial basis
      A2=xyz  x(T1)=x(x2-3r2/5)  y(T1)=y(y2-3r2/5)  z(T1)=z(z2-3r2/5)
          ksi(T2)=x(y2-z2)  eta(T2)=y(z2-y2)  zeta(T2)=z(x2-y2)

```

Data for 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**.

## 8.24 RENDOS (Renormalize partial DOS inside spheres)

This program renormalizes the partial DOS (from **tetra**) inside the spheres by a least squares procedure such that the sum of the PDOS is equal to the total DOS. As starting point it redistributes the interstitial DOS into atomic PDOS according to the inverse amount of the corresponding partial charge of the free atom inside its atomic sphere. A least squares procedure allows for a further readjustment due to localization/delocalization of the wave functions in the solid.

### 8.24.1 Execution

The program **rendos** requires **case.struct**, **case.outputst**, **case.int** and the partial density of states **case.dos1ev**, **case.dos2ev**, ... as input. You need to generate the total DOS and all partial DOS for ALL atoms and ALL relevant “chemical angular momenta” (eg. C-s and C-p; or Ti-s and Ti-d, but do not specify the total DOS of particular atoms) using the **tetra** program. Sometimes it is necessary to include also orbitals which are not occupied in the free atom (like Ti-4p).

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

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

You are asked interactively if you would like to optimize the fractional charges such that the sum of the partial DOS comes closer to the total DOS (to account for the missing interstitial). The renormalized DOS (**case.dosrn1ev**) can be plotted by **dosplot2.lapw -ren**.

## 8.25 SPAGHETTI (energy bandstructure plots)

This program generates an energy bandstructure plot (postscript file **case.spaghetti.ps** and xmgrace file **case.bands.gr**) 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.qt1** which can be obtained using **x lapw2 -qt1 -band**, see 7.9). 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 3.13 and 3.14). The circle size labelling and the header of the plot can easily be switched off by “header=0” in **case.insp**.

It is also possible to indicate the band masses using file **case.qtlmstar** which can be obtained using **x mstar** and **x mstarqt1**, see 8.18.

The file **case.spaghetti.ps** can be viewed by any postscript viewer (ghostscript, gv, ...). It is a plain text file and can be easily modified by an editor. For instance unwanted text can easily

be removed changing the bounding box pixels (x0,y0, xmax,ymax). **gv** will tell you the required pixels when you move the mouse to the desired left lower and right upper places, respectively.

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-clicking 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.

### 8.25.1 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.

### 8.25.2 Input

An example is given below:

```
----- top of file: case.insp -----
### Figure configuration
 5.0  3.0                                # paper offset of plot
10.0 15.0 3.0                            # xsize,ysize [cm], linebreak-parameter
 1.0  4                                # major ticks, minor ticks
 1.0  1  1                                # character height, font switch, header printing (0/1)
 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  1.0                       # jatom, jcol, size of heavier plotting
----- bottom of file -----
```

Interpretive comments on this file are as follows:

**line 1:** free format  
test

test                    line must start with '####'. Begin of figure description. This tests also if you use the new input (different from WIEN97 or early WIEN2k versions)

**line 2:** free format  
xoffset, yoffset

xoffset	x offset (in cm) of origin of plot
yoffset	y offset (in cm) of origin of plot

**line 3:** free format  
xsize,ysize,linebreak

xsize	plotsize in x direction (cm)
ysize	plotsize in y direction (cm)
linebreak	parameter to adjust the line break for non-continuous k-path

**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, header

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 defines.f
header	0/1 print header on top of plot and circle scale (bottom) (off/on)

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

linew	line width
ilin	0 dots or open circles
	1 lines
	2 lines and open circles
	3 lines and filled circles
icol	0 black
	1 one-color plot
	2 three-color plot
	3 multi-color plot
	4 multi-color plot,one color for each irred. representation

**line 7:** free format  
test

test	line must start with '###'. Begin of data description.
------	--

**line 8:** free format  
emin, emax, iunits

emin	energy minimum of plot
emax	energy maximum of plot

iunits		
	1	energies in Ry (internal scale)
	2	energies in eV with respect to $E_f$

**line 9:** free format

iferm, efermi

iferm	0	no line at EF
	1	solid line at EF
	2	dashed line at EF
	3	dotted line at EF
efermi		Fermi energy (Ry); can be found in the respective <b>case.scf</b> file. If set to 999., $E_f$ is not plotted (and iunits=2 cannot be used)

**line 10:** free format

nband1, nband2		lower and upper band index for bands which should show "band-character plotting" (if <b>case.qt1</b> is present and the proper switch is set, see below). In addition the corresponding x and y coordinates are written to file <b>case.spaghetti.ene</b> (which can be used for plotting with an external xy-plotting program).
-------------------	--	--

**line 11:** free format

jatom, jcol, jsize, power

jatom		If a <b>case.qt1</b> file is present, jatom indicates the atom whose character (selected by jcol) is used for "band-character plotting" (dots are replaced by circles with radii proportional to the corresponding weight, requires ilin=0,2,3). If set to zero or if <b>case.qt1</b> is not present, "band-character plotting" does not occur. If jatom is -1, the program will try to use <b>case.mstarqt1</b> , which should contain the corresponding band masses (see Section 8.18)
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 <b>case.struct</b> . (ignored for jatom=0). The description can be found in the header of <b>case.qt1</b> . jcol is ignored for band masses.
jsize		size factor for radii of circles used in "band-character plotting" ( $r = (QTL * jsize)^{power}$ )
power		(optional, default=1.0) size power factor, see above.

if **line 11** is repeated, one can average the QTLs for different atoms (but with identical jcol and jsize).

## 8.26 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/](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 **qt1** will do the job.

### 8.26.1 Execution

#### Execution

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

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

### 8.26.2 Input

TELNES3 requires one input file - **case.innes**. We recommend using **InnesGen<sup>TM</sup>** 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, 1 core)
285       (E-Loss of 1st edge in eV)
300       (energy of the incident electrons in keV)
0.0 20.0 0.1 (the energy mesh)
5.0 1.87   (collection semiangle, convergence semiangle, both in mrad)
10 1      (NR, NT, defining the integration mesh in the detector plane)
0.8       (spectrometer broadening in eV)
END
```

This first part of the file is not formatted and contains the following information:



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

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<sup>TM</sup>** 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  $NR^2 * NT$  points in total.

**L** a logarithmic grid with  $NR$  circles. The distance between circles increases exponentially. There are  $(2i - 1)NT$  points on circle  $i$ , and  $NR^2NT$  points in total. Circle  $i$  is at radius  $\theta_0 e^{(i-1)dx}$ , where  $dx$  depends on  $NR$ ,  $\alpha$  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.

#### ORIENTATION SENSITIVE

g1 g2 g3 (eg. 0.0 40.0 0.0) (no default value)

This key tells the program not to average over sample to beam orientations, but to use the particular sample to beam orientation defined by the three Euler angles (to be given in degrees). 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

type (eg. : q) (default : n)

The formula for the DDSCS contains an exponential factor in  $q$ , which we expand using the Rayleigh expansion. We identify each term in the expansion by the order  $\lambda$  of the spherical Bessel function  $j_\lambda(q)$  it contains. This key keeps some terms and discards others. This can be useful to eliminate unwanted transitions; to study a spectrum in greater detail; or simply to speed up the calculation significantly. Possible settings for 'type' are:

```
m : use lambda = 0 only
d : use lambda = 1 only
q : use lambda = 2 only
o : use lambda = 3 only
n : no selection rule, calculate all transitions
0-3 : all transitions up to lambda (eg., 1 means lambda = 0 and 1)
```

Be aware that the availability of the DOS limits the possible transitions (WIEN2k gives us the DOS only up to  $l=3$ ). In the nonrelativistic limit, the SELECTION RULE and LSELECTION RULE coincide - i.e., the  $\lambda = 1$  terms correspond to dipole transitions etc. This is no longer true in the relativistic case.

#### LSELECTION RULE

type (eg. : q) (default : d)

Whereas the previous key selects transitions by the order of the interaction potential, this key selects them by the L-character of the final states. Possible settings for 'type' are (the orbital momentum of the initial state being denoted with  $l$ ):

```
m : L=1
d : L=1 +/- 1
q : L=1 +/- 2
o : L=1 +/- 3
n : no selection rule, calculate all transitions
0-3 : |L-1| <= type
```

Be aware that the availability of the DOS limits the possible transitions (WIEN2k gives us the DOS only up to  $l=3$ ). In the nonrelativistic limit, the SELECTION RULE and LSELECTION RULE coincide – i.e., the  $\lambda = 1$  terms correspond to dipole transitions etc. This is no longer true in the relativistic case.

## EXTEND POTENTIALS

Rmax sampling lmax refine (e.g.: 3.0 15 0 1.0) (no defaults)

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

### 8.26.3 Practical considerations

A typical ELNES calculation consists of the following steps:

- ▶ initialize (`init_lapw`) and converge a SCF calculation (`run_lapw`)
- ▶ provide a suitable `case.innes` file
- ▶ if more excited states are needed than given by the SCF calculation, raise the upper energy limit in `case.in1` and run `x lapw1`
- ▶ create the `case.qt1` file using `x qt1 -telnes`
- ▶ calculate the EELS spectrum using `x telnes3`. It is generally a smart move to make the program calculate the DOS on the biggest energy grid you will ever need, save this to file, and simply read it from file for all future calculations (INITIALIZATION key). The same should be done for calculations using EXTEND POTENTIAL (use CORE WAVEFUNCTION key to save to file). This saves time. (In case of disk space problems, once the `case.qt1` file has been created, the `case.vector` files can be deleted. Similar, the `case.qt1` file can be deleted or compressed once the `case.dos` file exists.)
- ▶ add broadening to the spectrum using `x broadening`. If you wish, editing the `case.inb` file allows tweaking of the broadening.
- ▶ study the output (`case.elnes` or `case.broadspec` are the place to start).
- ▶ if you wish to do more calculations, save the current results using `save_eels -d calculation1`. Edit `case.innes` and run `x telnes3` again.

This sequence can conveniently be executed using `w2web` by simply clicking one button after the other.

### 8.26.4 Files

TELNES3 uses a lot of files. Many output files are only written if VERBOSITY is set to a high level. Many input files are required only for certain input settings in `case.innes`. We list here all files possibly used by TELNES3 (and listed in `telnes3.def`). Each filename is followed by "I" or "O" (input/output), a short description of the file content, and a comment on when the file is used.

- ▶ `case.innes` (I). Defines the ELNES calculation. Always read.
- ▶ `case.struct` (I). Defines the crystal. Always read.
- ▶ `case.vsp` (I). Spherical component of the crystal potential. Read unless core and final state wavefunctions are read from file.
- ▶ `case.vtotal` (I). Total crystal potential (can be generated by `lapw0`). Read if EXTEND POTENTIAL is used.
- ▶ `case.rotij` (I). Rotation matrices that transform q-vectors between equivalent atoms. Read if INITIALIZATION tells the program to do so.
- ▶ `case.dos` (IO). l-resolved density of states. Read or written depending on INITIALIZATION settings.

- ▶ `case.xdos` (IO).  $l_m, l'_m$ -resolved density of states. Read or written depending on `INITIALIZATION` settings; only if the calculation is orientation resolved.
- ▶ `case.qtl` (I). contains partial 'charge' components and Fermi energy. Read if DOS needs to be calculated (`INITIALIZATION`) or if Fermi energy is not specified using `FERMI`.
- ▶ `case.inc` (I). Specifies core states. Only read if core states are calculated.
- ▶ `case.kgen` (I). contains k-mesh to sample the Brillouin Zone. Read if DOS needs to be calculated.
- ▶ `case.outputelnes` (O). Main log file. Always written. Content depends on `VERBOSITY`.
- ▶ `case.elnes` (O). Total spectrum. Always written.
- ▶ `case.sdlm` (O). Partial  $(l, m)$  spectra. Written if `verbosity > 0`.
- ▶ `case.ctr` (O).  $(l, m, l'_m)$  crossterms. Written if `verbosity > 0` and calculation is orientation sensitive.
- ▶ `case.corewavef` (O). Contains core wavefunctions. Written if core wavefunctions were calculated and `verbosity > 1`.
- ▶ `case.final` (O). Contains APW radial basis functions for final states at selected energies. Written if `verbosity > 1`.
- ▶ `case.ortho` (O). Contains scalar products of initial and final states. Written if `verbosity > 1`.
- ▶ `case.matrix` (O). Proportionality between partial DOS and spectrum for each  $l$ -value. Written if `verbosity > 0` and `MODUS` is energy.
- ▶ `case.cdos` (O). Selected  $(l, m, l'_m)$  cross-DOS terms. Written if calculation is orientation sensitive and `verbosity > 1` or `INITIALIZATION` causes DOS to be written to file.
- ▶ `case.sp2` (O). Integrated cross sections as a function of collection angle for all  $l$ -values. Written if calculation is orientation sensitive, `MODUS` is set to angle and `verbosity > 1`.
- ▶ `case.angular` (O). Differential cross section as a function of scattering angle for all  $l$ -values. Written if calculation is orientation sensitive, `MODUS` is set to angle and `verbosity > 1`.
- ▶ `case.inb` (O). Settings for the broadening program. Always written.
- ▶ `case.eelstable` (O). Placeholder. Not currently used.
- ▶ `telnes3.def` (I). List of files used by `TELNES3`. Always read.
- ▶ `telnes3.error` (O). Error file containing current error message; empty after successful calculation. Always written.

## 8.27 TETRA (density of states)

This program calculates total and partial density of states (DOS) by means of the modified tetrahedron method [Blöchl et al., 1994]. Please note, the tetrahedron method will not work with just one k-point and `tetra` will automatically switch to a Gaussian broadening scheme (with default broadening of 0.01 Ry). 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 `qt1` 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 require `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 ( $l_m$ -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 qt1`

to generate the `case.qt1` and specified a specific coordinate system in `case.inq` (see Chapter 8.23).

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.

Using `KSElect=xx` in `case.int` you can plot a DOS with contributions only from tetrahedra containing k-point xx.

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).

### 8.27.1 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]
```

### 8.27.2 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)

### 8.27.3 Input

The required input file `case.int` can optionally be created using the `w2web` interface or the `configure.int.lapw` script (see 5.2.16).

An example is given below:

```
----- top of file: case.int -----
TiO2
-1.000 0.00250 1.200 0.003 # Title
 7 N 0.000 KSEL=-1 # EMIN, DE, EMAX for DOS, GAUSS-Broad
 0 1 tot # NUMBER OF DOS-CASES, N/G/L/B broadening, KSEL
 1 2 Ti-s # jatom, doscase, description
 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

**line 2:** free format

emin, delta, emax, broad

emin,	specifies the energy mesh (in Ry) where the DOS is calculated. (emin
delta,	should be set slightly below the lowest valence band; emax will be
emax	checked against the lowest energy of the highest band in <b>case.qt1</b> ,
	and set to the minimum of these two values; delta is the energy incre-
	ment.
broad	Gauss-broadening factor. Must be greater than delta to have any effect.

**line 3:** free format

ndos, Bmethod, broadening, KSEL

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 <b>case.dosX</b> 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)
KSEL	xx Create a DOS only from tetrahedra containing k-point number xx (in <b>case.klist</b> )
	-1 KSEL option switched off

**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 <b>case.struct</b> (see sec. 4.3); the respective description can be found in the header of <b>case.qt1</b> .
description	text used for further identification.

&gt;&gt;&gt;: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 <b>case.dossum</b> and <b>case.dossumev</b> .
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)

>>>:line 6 is repeated "nsum" times

## 8.28 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.qt1**. This file must be generated 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**.

### 8.28.1 Execution

#### Execution of the shell script xspec

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

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

#### Sequential execution of the programs

Besides calculating the X-ray spectra in one run using the **xspec** script, calculations can be done "by hand", i.e. step by step, for the sake of flexibility.

**initxspec** This program generates the appropriate input file **case.int**, according to the dipole selection rule, for the subsequent execution of the **tetra** program.

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

```
initxspec xspec.def or x initxspec [-up|-dn]
```

**tetra** The appropriate densities of states for (L+1) and (L-1) states respectively are generated by execution of the **tetra** program.

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

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

**txspec** This program calculates energy dependent dipole matrix elements. Theoretical X-ray spectra are generated using the partial densities of states (in the **case.doslev** file) and multiplying them with the corresponding dipole matrix elements.

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

```
txspec xspec.def or x txspec [-up|-dn]
```

**lorentz** The calculated spectra must be convoluted to account for lifetime broadening and for a finite resolution of the spectrometer before they can be compared with experimental spectra. In the **lorentz** program a Lorentzian is used to achieve this broadening. The program **lorentz** is executed by invoking the command:

```
lorentz xspec.def or x lorentz [-up|-dn]
```

If you want "orientation" sensitive XSPEC (like p-parallel and p-normal spectra, you may change in **case.int** the column-number to eg.  $p_x$  or  $p_z$  and rerun the last tree steps of the script above manually.

### 8.28.2 Dimensioning parameters

The following dimensioning parameters are collected in the files **param.inc** of **SRC.txspec** and **SRC.lorentz**:

IEMAX0	maximum number of energy steps in the spectrum (SRC.lorentz)
NRAD	number of radial mesh points
LMAX	highest l+1 in basis function inside sphere (consistent with input in case.in1)

### 8.28.3 Input

Two examples are given below; one for emission spectra and one for absorption spectra:

#### Input for Emission Spectra:

```
----- top of file: case.inxs -----
NbC: C K          (Title)
2                (number of inequivalent atom)
1                (n core)
0                (l core)
0,0.5,0.5        (split, int1, int2)
-20,0.1,3        (EMIN,DE,EMAX in eV)
EMIS             (type of spectrum, EMIS or ABS)
0.35             (S)
0.25             (gamma0)
0.3              (W)
AUTO             (generate band ranges AUTOMATICALLY or MANually)
-7.21            (E0 in eV)
-10.04           (E1 in eV)
-13.37           (E2 in eV)
----- bottom of file -----
```

#### Input for Absorption Spectra:

```
----- top of file: case.inxs -----
NbC: C K          (Title)
2                (number of inequivalent atom)
1                (n core)
0                (l core)
0,0.5,0.5        (split, int1, int2)
-2,0.1,30        (EMIN,DE,EMAX in eV)
ABS             (type of spectrum)
0.5             (S)
0.25            (gamma0)
----- bottom of file -----
```

Interpretive comments on these files are as follows.

**line 1:** free format

TITLE            Title

**line 2:** free format

NATO            Number of the selected atom (in case.struct file)

**line 3:** free format

NC              principle quantum number of the core state

**line 4:** free format

LC              azimuthal quantum number of the core state

The table below lists the most commonly used spectra:

Spectrum	n	l
$K$	1	0
$L_{II,III}$	2	1
$M_V$	3	2

Table 8.124: Quantum numbers of the core state involved in the x-ray spectra

**line 5** free format

SPLIT,  
INT1,  
INT2            split in eV between e.g.  $L_{II}$  and  $L_{III}$  spectrum (compare with the respective core eigenvalues), INT1 and INT2 specifies the relative intensity between these spectra. Values of 0, 0.5, 0.5 give unshifted spectra.

**line 6:** free format

EMIN,  
DE,  
EMAX            minimum energy, energy increment for spectrum, maximum energy; all energies are in eV and with respect to the Fermi level

EMIN and EMAX are only used as limits if the energy range created by the **lapw2** calculation (using the QTL switch) is greater than the selected range.

**line 7:** Format A4

TYPE	EMIS	X-ray emission spectrum
	ABS	X-ray absorption spectrum (default)

**line 8:** free format

S                broadening parameter for the spectrometer broadening. For absorption spectra S includes both experimental and core broadening. Set S to zero for no broadening.

**line 9:** free format

GAMMA0      broadening parameter for the life-time broadening of the core states.  
Set GAMMA0 to zero to avoid lifetime broadening of the core states.

**line 10:** free format

W              broadening parameter for the life-time broadening of valence states. Set  
W to zero to avoid lifetime broadening of the valence states.

**line 11:** format A4

BANDRA  
    AUTO      band ranges are determined AUTOMATICALLY (default)  
    MAN      band ranges have to be entered MANUALLY

**line 12:** free format

E0              Emission spectra: onset energy for broadening, E0, generated automati-  
cally if AUTO was set in line 10  
Absorption spectra: not used

**line 13:** free format

E1              Emission spectra: onset energy for broadening, E1, generated automati-  
cally if AUTO was set in line 10  
Absorption spectra: not used

**line 14:** free format

E2              Emission spectra: onset energy for broadening, E2, generated automati-  
cally if AUTO was set in line 10  
Absorption spectra: not used

---

# 9 Utility Programs

---

## Contents

---

9.1	add_columns	234
9.2	add_columns_new	234
9.3	afminput	234
9.4	animxsf	235
9.5	arrows	235
9.6	calla_pre	236
9.7	cif2struct	236
9.8	clminter	236
9.9	clmcopy	237
9.10	conv2prim	238
9.11	create_rho	238
9.12	eigenhess	239
9.13	eosfit	239
9.14	eosfit6	239
9.15	fleur2wien	239
9.16	hex2rhomb and rhomb_in5	240
9.17	join_vectorfiles	240
9.18	pairhess	240
9.19	patchsymm	242
9.20	plane	242
9.21	read_vorb_files	243
9.22	reformat	243
9.23	spacegroup	243
9.24	struct2cif	244
9.25	struct2poscar	244
9.26	structeditor	244
9.27	<b>StructGen</b> of <b>w2web</b>	246
9.28	supercell	246
9.29	symmetso	247
9.30	Tmaker	247
9.31	Visualization	247
9.32	xyz2struct	249
9.33	Unsupported software	250

---

## 9.1 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**.

## 9.2 add\_columns\_new

**add\_columns\_new** adds or subtracts two columns of data from two files You have to specify the filenames and cols 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**.

## 9.3 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 "KLASSENENGLEICH" (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. 4.5.5 on how to proceed in detail for AFM calculations and find further examples in **SRC.afminput**.

### 9.3.1 Execution

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

```
afminput afminput.def or x afminput
```

### 9.3.2 Dimensioning parameters

The following parameters are used:

NCOM	number of LM components in the density (in <b>param.inc</b> )
LMAX	max l for LM expansion of the density (in <b>param.inc</b> ).

## 9.4 animxsf

**animxsf** creates an animated xsf file of your position-optimization using **run/runsp -min** or **min\_lapw**. It calls a script **scfapos\_lapw** internally and generates **anim.tmp** from the scf file (using the :APOSxxx lines). Then it converts this into an animated xsf file **case.xsf**, which can be displayed using:

```
xcrysdn --xsf case.xsf # (Modify - AnimationControl - OK)
```

It is executed using:

```
x animxsf [-f case -pos]
```

The -pos switch will use :POS instead of :APOS (useful when using **min\_lapw**, but you need to copy **case.scf.mini** to **case.scf**).

## 9.5 arrows

This program was contributed by:

⇒ Evgeniya Kabliman  
 Institute for MaterialsChemistry  
 TU Vienna

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 **Xcrysdn** allows to display the “forces acting on all atoms” or the “differences between two structures” using arrows which indicate the movement of the atoms. The recommended sequence to visualize **forces** is:

- ▶ Prepare (copy) a struct and scf file with the initial structure using the names **case\_initial.struct** and **case\_initial.scf**.
- ▶ View **case\_initial.struct** in **Xcrysdn** and “File/Save as xsf-structure” with the name **case\_initial.xsf**.
- ▶ **x arrows**
- ▶ View the resulting **case\_forces.xsf** using: **xcrysdn --xsf case\_forces.xsf**. Switch on “Display/Forces” and adjust the length of the arrows in “Modify/Force-settings”.

while differences between the initial and relaxed structure can be viewed by:

- ▶ Prepare (copy) two struct files with the initial and the relaxed structure using the names: **case\_initial.struct** and **case\_final.struct**.
- ▶ View **case\_initial.struct** in **Xcrysden** and “File/Save as xsf-structure” with the name **case\_initial.xsf**.
- ▶ **x -delta arrows**
- ▶ View the resulting **case\_delta.xsf** using: **xcrysden --xsf case\_delta.xsf**. Switch on “Display/Forces” and adjust the length of the arrows in “Modify/Force-settings”.

## 9.6 calLa\_Pre

Small program to calculate lattice parameters at a given pressure using **case\_initial.struct** and **case.outputeos** (both are created by a standard wien2k volume optimization) at constants b/a and c/a ratio

Contributed by by Morteza Jamal(m.jamal57@yahoo.com).

## 9.7 cif2struct

**cif2struct** reads structural data in cif-format from **case.cif** and writes them into **case.struct**. It is executed using:

**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”; if necessary use **dos2unix** ).

Alternatively, **cif2struct** can work with **case.txt**, which contains the following data:

```
a                # a..Ang, b..Bohr
  0.0  0.0  0.0   # shift of origin
  4.7554 4.7554 12.991 90. 90. 120. # a,b,c,angles
'R-3c'          # spacegroup-symbol (see \STRUCTGEN{})
'Al'           # atom-name
0.0000000 0.0000000 0.3520000 # atomic position
'O'           # ...
0.3063000 0.0000000 0.2500000 # ...
...
```

## 9.8 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`
- ▶ optionally 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**.

## 9.9 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. 4.5.5 on how to proceed in detail for AFM calculations.

### 9.9.1 Execution

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

```
clmcopy clmcopy.def or x clmcopy
```

### 9.9.2 Dimensioning parameters

The following parameters are used in **param.inc**:

<b>NCOM</b>	number of LM components in the density
<b>NRAD</b>	number of radial mesh points
<b>NSYM</b>	number of symmetryoperations

### 9.9.3 Input

An example is given below:

```
----- top of file: case.inclmcopy -----
  2                                NUMBER of ATOMS to CHANGE
  1  2                            INTERCHANGE these ATOMS
-1.00000000000  0.00000000000  0.00000000000  SYMMETRY OPERATION
 0.00000000000 -1.00000000000  0.00000000000
 0.00000000000  0.00000000000 -1.00000000000
  0                                NUMBER of LM to CHANGE SIGN
  3  4                            INTERCHANGE these ATOMS
-1.00000000000  0.00000000000  0.00000000000  SYMMETRY OPERATION
 0.00000000000 -1.00000000000  0.00000000000
 0.00000000000  0.00000000000 -1.00000000000
  9                                NUMBER of LM to CHANGE SIGN
 1  0    1  0 -1.00
 3  0    3  0 -1.00
 3  2    3  2 -1.00
-3  2   -3  2 -1.00
 5  0    5  0 -1.00
 5  2    5  2 -1.00
-5  2   -5  2 -1.00
```

```

5 4      5 4 -1.00
-5 4     -5 4 -1.00
 1 0      0 0.50000
 0 1      0 0.00000
 0 0      1 0.50000

```

Interpretive comments on this file are as follows:

**line 1:** free format

NATOM            Number of atoms for which rules for copying the density will be defined

**line 2:** free format

N1, N2            Interchange spin-up and dn densities of atoms N1 and N2

**line 3-5:** free format

SYM              Symmetry operation for atom N1 to rotate into N2 (without translational part)

**line 6:** free format

NLM              Number of LM values, for which you have to change the sign when swapping up and dn-densities

**line 7ff:** free format

L1,M1,L2,M2,Fac    NLM pairs of L1,M1 (spin-up), which change into L2,M2 (spin-dn) and the respecting CLMs are multiplied by Fac

Lines 2-7ff have to be repeated NATOM times.

**line 8-10:** free format

SYM0             Symmetry operation (one of the operations of the NM-supergroup missing in the AFM-subgroup (transfers spin-up into spin-dn atom))

## 9.10 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**


## 9.11 create\_rho

The program **create\_rho** is called by the script **create\_elf\_lapw** and creates  $\alpha=(\tau-\tau W)/\tau_{TF}$ ,  $z=\tau W/\tau$  or  $ELF=1/(1+\alpha^2)$  from the corresponding rho, rho\_onedim or xsf files containing the different kinetic energies tau, tauW and tauTF. See chapter 5.10.13.

The sources of the program **create\_rho.f** are supplied in **SRC.trig**.

## 9.12 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 can give a hint about a dynamically unstable structure (imaginary frequencies). Some more description is given in `$WIEN-ROOT/SRC_pairhess/README`.

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

```
x eigenhess
```

## 9.13 eosfit

Small program to calculate the Equation of States (EOS; Equilibrium volume  $V_0$ , Bulk modulus  $B_0$  and it's derivative  $B'_0$ ). The Murnaghan [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 `epplot_lapw` using `case.analysis` (see 5.10.1 and 5.3.1).

The sources are supplied in `SRC.eosfit`.

## 9.14 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`.

## 9.15 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
```

## 9.16 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**.

## 9.17 join\_vectorfiles

This program was contributed by:

⇒ Philipp Wissgott  
 Institute of Solid State Physics  
 TU Vienna

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

Interactive program to combine parallel vector and energy files (`case.vector_xx`, `case.energydum_xx` and `case.energy_xx`) into single files (`case.vector`, `case.energydum` and `case.energy`).

Executed by:

- ▶ x joinvec [-up/-dn/-so/-hf/-enefiles/-band]
- ▶
- ▶ (x join\_vectorfiles [-up/-dn/-so/-hf/-enefiles] # for compatibility

## 9.18 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 strength 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.

### 9.18.1 Execution

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

```
pairhess pairhess.def or x pairhess [-copy]
```

The switch `-copy` copies `.minpair` to `.minrestart` and `.min.hess`, which are needed in `min_lapw`.

### 9.18.2 Dimensioning parameters

The following parameters are used in `param.inc`:

```
NATMAX    max. number of atoms)
NEIGMAX    max number of neighbours
```

### 9.18.3 Input

`pairhess` uses an optional input file `case.inpair`, which is needed only for an experienced user for better tailoring of certain default parameters.

An example is given below:

```
----- top of file: case.inpair -----
10.0 2.0 0.25 (Rmax, Decay, ReScale)
0.05 1.0 0   (Cutoff, Diag, mode)
0.2         (ZWEIGHT
```

Interpretive comments on this file are as follows:

#### line 1: free format

RMAX	Maximum distance (a.u.) for considering neighbors. 8-12 is good.
DECAY	Exponential decay applied to neighbors when calculating the pairwise bond 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 <code>.min.hess</code> from a previous run (adjust until numbers for similar multiplicities are similar), or by adjusting the frequencies (see also <code>eigenhess</code> ).

**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.
---------	---

## 9.19 patchsymm

This program was contributed by:

⇒ James Rondinelli, Bin Deng and Laurence Marks  
 Dept. Materials Science and Engineering  
 Northwestern University  
 Evanston, USA  
 l-marks@northwestern.edu

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

This program performs a symmetry check on the positions and produces a new struct file **case.struct.new**. It is useful in case something went wrong during **min.lapw** (rounding errors of positions) or the cif/amc file did not have enough digits (eg. "1/3" was prepresented by "0.33333" only). The file case.outputpatch gives information on how parameters changed.

### 9.19.1 Execution

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

```
patchsymm patchsymm.def or x patchsymm
```

## 9.20 plane

**plane** helps to generate **case.in5** for density plots with **lapw5** (for orthogonal and hex lattices only). The plane will be specified by 3 atoms and you need an auxiliary file **plane.input**, which contains:

```

a,b,c      # lattice parameters
x0,y0,z0   # position of atom (fractional coordinates), which will be centered in the plot
x1,y1,z1   # position of atom, which will be ``below'' the centered atom
x2,y2,z2   # position of atom, which will show to the ``left''
x1,y1      # lenght (in bohr) of plot in x and y direction.
'P'       # defines lattice, either P (cartesian coordinates) or H (hexagonal) supported

```

The source of this program is supplied in **SRC.trig**.

## 9.21 read\_vorb\_files

**read\_vorb\_files** adds **case.vorbup/dn\_Bext** to **case.vorbup/dn**. It is called by **runsp\_lapw** when both, an external field and LDA+U/EECE is used.

```
read_vorb_files -up / -dn
```

The source of this program is supplied in **SRC.trig** and was contributed by William Lafargue-Dit-Hauret (Univ. Rennes).

## 9.22 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**.

## 9.23 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**.

## 9.24 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-immn.fr) and L.D.Marks (Lmarks@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.

## 9.25 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 packages **dftd3** and **dftd4** when periodic boundary conditions are switched on or off, respectively. It is executed using:

```
x struct2poscar or struct2poscar struct2poscar.def
```

## 9.26 structeditor

This program was contributed by:



Robert Laskowski  
 email: rolask@ihcp.a-star.edu.sg  
 Please make comments or report problems with this program to the WIEN-mailinglist. If necessary, we will communicate the problem to the authors.

This package helps to manipulate structures. Usually one would start from an appropriate (simple) **case.struct** file, and this tool allows to add or manipulate atoms (with or without symmetry considerations), or generate arbitrary supercells or surfaces. It is commandline driven and targeted for the more experienced user, who “knows what he wants to do” and is just looking for a convenient tool.

It consists of a couple of octave (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 * convestrs structure into the conventional form
```



```

makeprimitive      * converts structure to the primitive form
makesupercell     * creates supercell
makesurface       * 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 middle)
rescale_c_2       * rescales c for surface cell (vacume above)
rescale_c_3       * rescales c for surface cell (vacume audside)
rmatom            * removes an atom
rmeqatoms         * 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
showstruct        * 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.

### 9.26.1 Execution

The **structeditor** is invoked within the octave environment and a typical sequence of commands could be:

```

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

```

## 9.27 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 9.23 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.  $-y$ ,  $x, -z + 1/2$ ).

## 9.28 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  $45^\circ$  like  $\sqrt{2} \times \sqrt{2}$  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, eg. you may want to create a surface slab such that it is centered around  $z=0.5$  (and not  $z=0$ ), so that plotting programs (xcrysden) produce nicer pictures of the structure.

For the experienced user a much more flexible (but also more complicated) tool is available, namely the **structeditor** package (see Sect.9.26).

*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, ....*

### 9.28.1 Execution

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

**supercell** or **x supercell**

## 9.29 symmetso

This program helps to setup spin-orbit calculations in magnetic systems. Since SO may break symmetry in certain spacegroups, it classifies your symmetry operations into operations **A**, which do not invert the magnetization (identity, inversion, rotations with the rotation axis parallel to magnetization), **B**, which invert it (mirror planes) and **C**, which change the magnetization in some other way. (Note: magnetization is a result of a circular current, or equivalently, an axial vector resulting from a vector product  $\hat{z} \sim \hat{x} \times \hat{y}$ ). **symmetso** will keep all A- 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\_orb** file with proper symmetry and equivalent atoms. In addition proper input files **case.in1**, **case.in2**, **case.inc**, **case.vspup/dn**, **case.vnsup/dn**, **case.clmsum**, **case.clmup/dn**, **case.r2vup/dn**, **case.tausum/up/dn**, **case.dmatup/dn** are generated, so that you can continue with **runsp-so** without any further changes.

However, please note that for certain cases (transformation from cubic to non-cubic atoms, changes of the local rotation matrix), the non-spherical potential (**case.vnsup/dn**) is not fully correct inside the spheres and self-consistency is necessary. This is in particular important when calculating the magnetocrystalline anisotropy using the force theorem (non-selfconsistent calculation of the change of eigenvalues for different directions of magnetization). For these type of calculations it is recommended to do all calculations (with and without SO) with a common low symmetry (possibly even P1) struct file.

### 9.29.1 Execution

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

```
symmetso symmetso.def or x symmetso [-c]
```

Usually it is called from the script **init\_so\_lapw** and thus needs not to be invoked manually.

## 9.30 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).

## 9.31 Visualization

### 9.31.1 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 9.5)

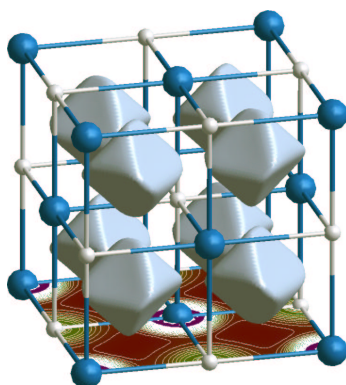


Figure 9.1: 3D electron density in TiC generated with XCrysDen

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/>

### 9.31.2 VESTA

**VESTA** (Visualization for Electronic and STructural Analysis) written by Koichi Momma and Fujio Izumi is a very nice and flexible visualization program of crystal structures. It can read **case.struct** and **case.cif** files, edit structures, write modified cif-files, display arrows and polyhedra. It can also read **case.xsf** files and thus display electron densities calculated by **3ddens**.

VESTA is available from:

<http://jp-minerals.org/vesta/en/>

### 9.31.3 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/>

## 9.32 xyz2struct

**xyz2struct** reads the cell parameters and position of atoms from a xyz/POSCAR file and then creates the corresponding **xyz2struct.struct**. Three types of xyz/POSCAR files can be read by **xyz2struct**:

- ▶ 1st type:

```
Mo Te
1.0000000000000000 Ang
 3.5473923118306230 0.0000000000000000 0.0000000000000000
-1.7736961559153115 3.0721318592349283 0.0000000000000000
 0.0000000000000000 -0.0000000000000000 18.6101999999999990
 1 2
Cartesian
0.0000000000000000 0.0000000000000000 9.3050999999999995
1.7736961559153113 1.0240439530783094 11.1101999999999990
1.7736961559153113 1.0240439530783094 7.5000000000000000
```

1st line: Symbols of the groups of atoms.

2nd line: Scaling factor of the lattice vectors and (optional) units of the lattice vectors and position of atoms ("Ang" or "Bohr"). If no units are specified, then Å units are assumed.

3rd line: Cartesian coordinates of the lattice vector a.

4th line: Cartesian coordinates of the lattice vector b.

5th line: Cartesian coordinates of the lattice vector c.

6th line: Number of atoms in each groups (as specified in the 1st line).

7th line: Type of coordinate system that is used for the position of atoms: "Cartesian" or "Direct" (or equivalently "Fractional").

8th and following lines: Position of atoms.

- ▶ 2nd type (Materials Project, <https://materialsproject.org/>):

```
MoS2
1.0 Ang
3.190316 0.000000 0.000000
-1.595158 2.762894 0.000000
0.000000 0.000000 14.879004
Mo S
2 4
Direct
0.333333 0.666667 0.250000 Mo
0.666667 0.333333 0.750000 Mo
0.666667 0.333333 0.355174 S
0.333333 0.666667 0.855174 S
0.666667 0.333333 0.144826 S
0.333333 0.666667 0.644826 S
```

- 1st line: Header (ignored).
- 2nd line: Scaling factor of the lattice vectors and (optional) units of the lattice vectors and position of atoms (“Ang” or “Bohr”). If no units are specified, then Å units are assumed.
- 3rd line: Cartesian coordinates of the lattice vector **a**.
- 4th line: Cartesian coordinates of the lattice vector **b**.
- 5th line: Cartesian coordinates of the lattice vector **c**.
- 6th line: Symbols of the groups of atoms.
- 7th line: Number of atoms in each groups (as specified in the 6th line).
- 8th line: Type of coordinate system that is used for the position of atoms: “Cartesian” or “Direct” (or equivalently “Fractional”).
- 9th and following lines: Position of atoms (the symbols of the atoms are ignored).

► 3rd type (Computational 2D Materials Database (C2DB), <https://cmr.fysik.dtu.dk/c2db/c2db.html>):

```

3
Lattice="3.54739231183062 0.0 0.0 -1.77369615591531 3.07213185923493 0.0 0.0 0.0 18.6102" ...
Mo      0.00000000  0.00000000  9.30510000  5.75973452
Te      1.77369616  1.02404395  11.11020000 -3.32538427
Te      1.77369616  1.02404395  7.50000000 -0.00000000

```

- 1st line: Number of atoms.
- 2nd line: Cartesian coordinates (in Å) of the lattice vectors (inside the quotation marks after the keyword “Lattice”).
- 3rd and following lines: Symbol and cartesian coordinates (in Å) of the atoms.

**xyz2struct** is executed with either (the xyz/POSCAR file has to be **case.xyz**)

```
x xyz2struct [-settol 1e-5]
```

or (the xyz/POSCAR file **file** can have any name)

```
xyz2struct < file [1e-5]
```

where settol (set to 1e-5 by default) determines the rounding of the position of atoms.

### 9.33 Unsupported software

On our website [http://www.wien2k.at/reg\\_user](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 [peter.blaha@tuwien.ac.at](mailto:peter.blaha@tuwien.ac.at) and we will add it to this page.

---

# 10 How to run WIEN2k for selected samples

---

Three test cases are provided in the **WIEN2k** package. They contain the two starting files **case.struct** and **case.inst** and all the output so that you can compare your results with them.

The test cases are the following (where the names correspond to what was called CASE in the rest of this User's Guide)

**TiC**  
**Fccni**  
**TiO2**

We recommend to run these test cases (in a different directory) and compare the output to the provided one. All test cases are setup such that the CPU-time remains small (seconds). For real production runs higher precision is recommended (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** where 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  $[\text{Ar}] 3d^8, 4s^2$ . We treat the  $1s, 2s, 2p$  and  $3s$  as core states, and  $3p$  (as local orbital),  $3d, 4s$  and  $4p$  are handled as valence states. In a spin-polarized calculation the file structure and the sequence of programs is different from the non-spin-polarized case (see 4.5.2).

Create a new session and its corresponding directory. Generate the structure with the following data (we can use a large sphere as you will see from the output of **nn**):

Title	fcc Ni
Lattice	F
a	6.7 bohr
b	6.7 bohr
c	6.7 bohr
$\alpha, \beta, \gamma$	90
Atom	Ni, enter position (0,0,0) and RMT (3 % reduction)

Initialize the calculation using spin-polarization and the default precision. It leads to RKmax=7.5 and a  $17^3$  k-mesh. (A ferromagnetic metal needs many k-points to yield reasonably converged magnetic moments).

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 = -0.03288
:MMI001: MAGNETIC MOMENT IN SPHERE 1 = 0.66671
:MMTOT: TOTAL MAGNETIC MOMENT IN CELL = 0.63383
```

Save the calculation using **save\_lapw prec1**.

Make another initialization with **-prec 2** and **-nodstart**. Since this is now a very dense k-mesh and the calculation will take some time, we run the scf cycle in k-parallel mode (**runsp\_lapw -p -cc 0.0001**) with the following **.machines** file (assuming we have a 8 core machine):

```
1:localhost
1:localhost
1:localhost
1:localhost
omp_global:2
```

The resulting moments are now very well converged and change to:

```
:MMINT: MAGNETIC MOMENT IN INTERSTITIAL = -0.03331
:MMI001: MAGNETIC MOMENT IN SPHERE 1 = 0.68115
:MMTOT: TOTAL MAGNETIC MOMENT IN CELL = 0.64784
```

Save the calculation under the name **prec2**.

## 10.3 Rutile ( $TiO_2$ )

This example shows you how to “optimize internal parameters” and do a k-point parallel calculation in 3 different ways.

### 10.3.1 using w2web and the recommended MSR1a minimization

Create a new session and its corresponding directory. Generate the structure with the following data (reduce the RMT by 3 %. Note: the O sphere will be made smaller than that of Ti because Ti-d states are harder to converge than O-p):

Title	TiO2
Spacegroup	$P4_2/mnm$ (136)
a	8.682 bohr
b	8.682 bohr
c	5.592 bohr
$\alpha, \beta, \gamma$	90
Atom	Ti, enter position (0,0,0)
Atom	O, enter position (0.3,0.3,0)



**StructGen** will automatically add the equivalent positions.

Initialize the calculation using `-prec 1n` ("n" because we know it is an insulator). It will set `RKmax=6.17` and produce 9 k-points in the IBZ.

If you have more cpus available (a multi-core machine or, very efficiently, couple several PCs with a common NFS filesystem to your own cluster, 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 (or use 3 lines with `1:localhost`; we use 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.

During the scf-cycle monitor `tio2.dayfile` and check convergence (:ENE, :DIS, :FGL002), either using "Utils/Analysis" in **w2web**. You should see some convergence of :FGL002 and then a big jump in the final cycle, when the valence-force corrections are added. Only the last force (including this correction) is valid.

Since this force is quite large, you should first "save" the calculation ("`starting_structure`") and then optimize the position of the O-atom:

Start the structure minimization in **w2web** using "Execution  mini.positions". This will generate `TiO2.inM`, and you should use the recommended procedure using "**MSR1a**". Follow the instructions.

Watch the minimization (:ENE, :FR, :FGL002, :POS002) using the file `TiO2.scf`, which contains the history of all steps (see also Sec.5.3.2). Since we change positions and density simultaneously, it will take more steps and the forces may oscillate a lot, but as long as :ENE goes down, it is fine. Once the forces are 3 times below TOLF, MSR1a mode will be switched to MSR1 and a normal scf cycle will continue until the convergence criteria are met.

The final structural parameter of the O-atom should be close to  $x=0.304$ , which compares well with the experimental  $x=0.305$ .

**Save relaxed.structure.**

### 10.3.2 PORT minimization

Alternatively you can also try option PORT with `tolf=1.0` (instead of 2.0), otherwise stay with the default parameters. Create a new session (and case) and repeat the steps above. Then start the minimization.

This will create `TiO2.inM` automatically, call the program `min`, which generates a new struct file using the calculated forces, and continues with the next scf cycle. It will continue until the forces are below 1 mRy/bohr (`TiO2.inM`). The final results are not "saved" automatically but can be found in the "current" calculation.

You should watch the minimization (:ENE, :FGL002, :POS002) using the file `TiO2.scf_mini`, which contains the final iteration of each geometry step (see also Sec.5.3.2). If the forces in this file oscillate from plus to minus and seem to diverge, or if they change very little, you can edit `TiO2.inM` (change the method, reduce or increase the stepsize), and remove `TiO2.tmpM` (contains the "history" of the minimization and is used to calculate the velocities of the moving atoms). (This should not be necessary for the rutile example, but may occur in more complex minimizations. See comments in Sec. 5.3.2).

### 10.3.3 Command line usage:

Once you gain experience, the command line interface of WIEN2k is much more powerful and faster to use. Here we provide a first example:

```

mkdir TiO2; cd TiO2      # create a new directory and change into it.
makestruct              # enter the necessary data similar as above
cp init.struct TiO2.struct
init_lapw -prec ln
cp $WIENROOT/SRC_templates/.machines .      # and edit this file accordingly
run_lapw -p -fc 1          # scf cycle with fixed positions
save_lapw starting_structure
run_lapw -p -min -fc 1 -cc 0.0001          # scf cycle optimizing positions (MSR1a)
save_lapw relaxed_structure

```

Now analyze the results using the Linux command **grep** and a WIEN2k aliases **grepline**:

```

grep :ENE relaxed_structure.scf          # check convergence
grep :FGL002 relaxed_structure.scf
grep :POS002 relaxed_structure.scf
grepline :ENE '*scf' 1                   # compare original and relaxed structure
grepline :FGL002 '*scf' 1
grepline :POS002 '*scf' 1

```

## 10.4 Supercell calculations on TiC

This example shows you how to create a supercell of TiC, which could be used to simulate a TiC-surface or vacancies, impurities or core-holes for X-ray absorption / ELNES spectroscopy. I'll describe the procedure using Unix and WIEN2k commands in an xterm, but of course you can do the same in **w2web**.

Create a new directory, copy the original TiC struct file into it and run **supercell** program:

```

mkdir super
cd super
cp ../TiC/TiC.struct .
x supercell

```

Specify "TiC.struct", a "2x2x2" supercell, "F" lattice (this will create a cell with 16 atoms, you can also create 32 or 64 atom cells using B or P lattice type. Note: surfaces require a P supercell).

```
cp TiC_super.struct super.struct
```

and edit this file to make some changes. You could eg.

- ▶ delete an atom (to simulate a vacancy)
- ▶ replace an atom by another element (impurity)
- ▶ "label" an atom (put a 1 in the 3rd column next to the element name) to make this atom unique (needed eg. for core-holes)
- ▶ displace an atom (for phase transitions or phonons)

*Note: it is important to make at least one of these changes. Otherwise the initialization will restore the original unit cell (or the calculations will fail later on because symmetry is most likely not correct)*

In our example we will simulate a **C vacancy** and remove the last C atom (number 16) and change the total number of atoms (2nd line) to 15.

- ▶ **x nn.** You will see a complain about equivalent atoms and a new struct file has been generated with only 4 non-equivalent atoms. Accept the automatically generated struct file (**cp super.struct\_nn super.struct**) and continue.
- ▶ **x sgroup.** Inspect **super.outputsgroup**. You will see that it complains about this structure and a new one has been generated. Compare **super.struct** and **super.struct\_sgroup**. What has been changed ? Accept the automatically generated struct file (**cp super.struct\_sgroup super.struct**).
- ▶ **x symmetry.** Inspect **super.outputs**. As expected, there are no problems anymore, we have produced a struct file with the correct symmetry and can continue with the initialization. Note that atom 2 has point group 4/mm and a non-trivial "local rotation matrix", which rotates the axes such that the 4-fold rotation axis becomes z. In addition, this site has a free position parameter and the Ti atoms can move towards/away from the C vacancy.
- ▶ **init\_lapw**

For a "core-hole" calculation you would "label" the last atom as "C 1" and run the same steps as above. Then edit **super.inc** and remove one core electron from the desired atom and state (1s from the "C 1" atom). 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 in the workshops section:

<http://www.wien2k.at/events>,

in particular in the exercises description at the latest workshops



## **Part III**

# **Installation of the WIEN2k package and Dimensioning of programs**



---

# 11 Installation and Dimensioning

---

## Contents

---

11.1 Requirements . . . . .	259
11.2 Installation of WIEN2k . . . . .	262
11.3 w2web . . . . .	266
11.4 Environment Variables . . . . .	268

---

## 11.1 Requirements

**WIEN2k** is written in FORTRAN 90 and requires a Linux/UNIX operating system since the programs are linked together via C-shell scripts. It has been implemented successfully on Intel or AMD based computer systems, starting from Laptops, PCs, workstations and high performance clusters running under Linux, (IBM RS6000), and on Macs. Hardware requirements will change from case to case (small cases with 60 atoms per unit cell can be run on almost any PC/laptop under Linux), but we recommend a more powerful multi-core (“latest generation”) Intel-I7 (I9) PC with at least 32 GB) memory and plenty of disk space (1-2 TB).

For OpenMP and coarse grain parallelization on the k-point level, a cluster of (fairly cheap) PCs with Gb Ethernet is sufficient. Faster communication (Infiniband) is needed for the fine grain (single k-point) mpi-parallel version and typical then expensive Intel Xeon nodes are used. Note, that a good Intel I7 (I9) processor often has a better single core performance than expensive Xeons nodes, and thus has (at least for small/medium sized cases) the far best price/performance ratio.

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. For Intels Parallel Studio add some thing like:

```
source /opt/intel/.../bin/compilervars.sh intel64
```

to your **.bashrc** file or for Intels oneAPI:

```
source /opt/intel/oneapi/setvars.sh
```

When you have the necessary hardware, you can also install the mpi-version and the most easy compilation is with Intel-mpi (included in Intels oneAPI) or opnMPI.

A free but decent alternative for compiler and library is using **gfortran** (at least version 6 or higher) and the **OpenBlas**-library, which should be available in any modern Linux distribution. In addition OpenMPI or mpich is necessary for the mpi-version.

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

- ▶ ifort+mkl or gfortran+openblas
- ▶ FFTW3 (mandatory since WIEN2k\_21 for both, sequential and fine grain mpi-parallelization)
- ▶ perl 5 or higher (for **w2web** only)
- ▶ tcsh
- ▶ Linux tools like gcc, bc, make, awk
- ▶ emacs or another editor of your choice (vi, gedit, ...)
- ▶ ghostscript (with jpeg support)
- ▶ gnuplot (with png support)
- ▶ www-browser
- ▶ pdf-reader (okular, evince, xpdf, ...)
- ▶ XCrysDen for visualization, highly recommended
- ▶ VESTA for visualization, highly recommended
- ▶ Tcl/Tk-Toolkit (for Xcrysden only)
- ▶ Xmgrace (only for optional plotting)
- ▶ Octave (only for the **structeditor**)
- ▶ Python + numPy (only for **BerryPI** and **mstar**)
- ▶ MPI (ONLY for fine grain mpi-parallelization, Intel-MPI, OpenMPI or mpich)
- ▶ SCALAPACK (ONLY for fine grain mpi-parallelization)
- ▶ ELPA (ONLY for fine grain parallelization, optional, but highly recommended since ELPA is 2 times faster than SCALAPACK)
- ▶ LIBXC (ONLY when using special, non-standard DFT approximations or gKS (scf) meta-GGA calculations or for the stress tensor)

The script **check\_minimal\_software\_requirements.sh** checks if it can find the necessary (and optional) software.

Usually these packages should be available on modern systems. If one of these packages is not available it can either be installed from the Linux distribution, 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 some packages are given below.

### 11.1.1 Installation tips for fftw3, ELPA and LIBXC

Most importantly, fftw3, ELPA and LIBXC must be installed with the same FORTRAN compiler that you will use for WIEN2k. These libraries may be already available in your Linux distribution, but most likely ONLY work with **gfortran**, not with Intels **ifort**.

#### ELPA

ELPA is ONLY necessary if you plan to use the mpi-version of WIEN2k (and have the corresponding hardware). It is NOT necessary for an installation on a single PC. The following should install ELPA, assuming that you have ifort and mkl (Intels OneAPI), OpenMPI and use the default gcc compiler.

- ▶ cd "elpa-directory"
- ▶ git clone https://github.com/marekandreas/elpa
- ▶ autogen.sh



- ▶ `configure FC=mpifort SCALAPACK.LDFLAGS="-L$MKLRROOT/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -lmkl_blacs_openmpi_lp64 -lpthread -lm -ldl -liomp5 -Wl,-rpath,$MKLRROOT/lib/intel64" SCALAPACK.FCFLAGS="-L$MKLRROOT/lib/intel64 -lmkl_scalapack_lp64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -lmkl_blacs_openmpi_lp64 -lpthread -lm -I$MKLRROOT/include/intel64/lp64" --prefix=/pathname/elpa-2022.05.001 FCFLAGS=-O3 CFLAGS=-O3 -mfma -funsafe-loop-optimizations -funsafe-math-optimizations -ftree -vect-loop-version -ftree-vectorize`  
Optionally you could add: `--enable-avx512` or/and `--enable-openmp` when you have the corresponding hardware,  
specify the proper mpi-compiler name: `FC=mpifort`,  
where `pathname` is the path you want to install `elpa` (when you are root eg. `/opt`; otherwise eg. your home-directory), you will have to specify this path again in the `LDFLAGS`),  
replace `-lmkl_blacs_openmpi_lp64` by `-lmkl_blacs_intelmpi_lp64` if you use Intel-mpi.
- ▶ `make`
- ▶ `make install`

Please note, the ELPA interface has changed for ELPA-versions 2017 and later. Both interfaces are included in WIEN2k, but you have to select `-DELPA` (newer versions) or `-DELPA15` in the Makefile of `lapw1` (or during `siteconfig_lapw`). Please refer to the ELPA documentation for additional information regarding openMP support and other options!

## fftw

fftw is now mandatory, both for the sequential and mpi-parallel version.

- ▶ Download the `fftw-3.3` sources from <http://www.fftw.org/download.html>
  - Please note, the `fftw-3.x` versions are incompatible with `fftw-2.x` and `fftw-2` is no longer supported in WIEN2k
- ▶ unzip and untar the downloaded file
- ▶ Change into the expanded directories and configure the compilation.
  - Define your fortran compiler in the variables `F77` (`setenv F77 ifort`, or `export F77=ifort`)
  - Use `./configure --prefix=/pathname --enable-openmp` to configure compilation.
  - If you want to use also the mpi-version of WIEN2k, add the `--enable-mpi` switch to the line above.
- ▶ `make`
- ▶ `make install` (if you specified a "system-directory" like `/usr/local` you must have proper permissions for this step, eg. become root user)

Optionally, one can also use in the sequential (non-mpi) version of `lapw0` and `lapw2` the `MKL-fftw3` routines, not the self-compiled `fftw3-binaries`, which are a bit fister. This may speedup the `fft`-parts of these programs a bit, but overall it is probably not worth the effort, in particular since one cannot use the `mkl`-version with `mpi`.

## Installation of LIBXC

In order to use the library of exchange and correlation functionals **LIBXC** [Marques et al., 2012, Lehtola et al., 2018], you need to install it and recompile `lapw0`. **LIBXC** is required for both, `gKS` meta-GGA calculations and the stress tensor. The steps are the following:

- ▶ Download **LIBXC** (the most recent version, at least 5.2.3), from <https://libxc.gitlab.io/>

- ▶ Compile **LIBXC** with the same compiler that was used to compile **WIEN2k**. The basic steps are the following (example with ifort):

```

- cd libxc-6.2.2/
- ./configure FC=ifort --prefix=$LIBXCDIR
- make
- make check
- make install

```

where LIBXCDIR is the directory where you choose to install 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)/lib64/ -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 Check the software requirements

The script **check\_minimal\_software\_requirements.sh** checks if it can find the necessary (and optional) software. PLEASE NOTE: It does not make sense to continue when NECESSARY software (tcsh, fortran compiler (ifort or gfortran), FFTW3, make, bc) is not on your system.

### 11.2.2 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**, although they might not be up-to-date.

Uncompress and expand all files using:

```

tar -xvf wien2k_XX.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_3ddens
SRC_afminput
SRC_afmsim
SRC_aim
SRC_animxf
SRC_arrows
SRC_balsac-utils
SRC_BerryPI

```

```
SRC_broadening
SRC_cif2struct
SRC_clmaddsub
SRC_clmcopy
SRC_dipan
SRC_dstart
SRC_elast
SRC_eosfit
SRC_eosfit6
SRC_filtvec
SRC_fsgen
SRC_Globals
SRC_hf
SRC_initxspec
SRC_IRelast
SRC_irrep
SRC_joint
SRC_kgen
SRC_kram
SRC_lapw0
SRC_lapw1
SRC_lapw2
SRC_lapw3
SRC_lapw5
SRC_lapw7
SRC_lapwdm
SRC_lapwso
SRC_lcore
SRC_lib
SRC_lorentz
SRC_lstart
SRC_mini
SRC_mixer
SRC_mstar
SRC_nlvdw
SRC_nmr
SRC_nn
SRC_optic
SRC_optimize
SRC_orb
SRC_pairhess
SRC_pes
SRC_phonon
SRC_qt1
SRC_reformat
SRC_rendos
SRC_sgroup
SRC_spacegroup
SRC_spaghetti
SRC_structeditor
SRC_sumhpara
SRC_sumpara
SRC_supercell
SRC_symmetry
SRC_symmetso
SRC_telnes3
SRC_templates
SRC_tetra
SRC_Tmaker
SRC_trig
SRC_txspec
SRC_usersguide_html
SRC_vecpratt
SRC_w2w
SRC_w2web
SRC_wplot
example_struct_files
Tic
TiO2
fccni
```

Thus, each program has its source code (split into several files) in its own subdirectory. All programs are written in FORTRAN90 (except SRC\_sgroup and SRC\_reformat, which are in C).

**/SRC** contains the users guide (in form of a postscript file **usersguide.ps** and as pdf-file **usersguide.pdf**), all c-shell scripts and some auxiliary files.

**/SRC.usersguide.html** contains the html version of the UG.

`/Fccni`, `/TiC` and `/TiO2` contain three example inputs and the respective outputs.

`/example_struct_files` contains a collection of various struct files, which could be of use especially for the less experienced user.

`/SRC_templates` contains various input templates.

In addition to the expansion of the tar-files `./expand_lapw` copies also all csh-shell scripts from `/SRC` to the current directory and creates links for some abbreviated commands.

### 11.2.3 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 `WIEN2k_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; we support directly `ifort+mkl` (with/without SLURM) and `gfortran+OpenBlas`.
- ▶ your FORTRAN90 and C compilers;
- ▶ your compiler and linker options as well as the place for LAPACK and BLAS libraries. For the standard systems we have included some recommended compiler and linker options, which are known to work on our systems and you can probably accept all recommendations ( see also sec. 11.2.5). This generates Makefiles from the corresponding `Makefile.orig` in all subdirectories.
- ▶ configuration of parallel execution will ask whether your system is shared memory only (use this ONLY if you have only ONE shared memory computer), 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 `ssh`.
- ▶ You will then be asked whether you want to run fine-grained mpi-parallel. This is only possible if FFTW, MPI, SCALAPACK (both are included in Intels openAPI) and, optionally, ELPA are installed on your system and requires a fast network (infiniband) or a large shared memory machine. It pays off only for bigger cases (more than 30 atoms, matrixsize > 5000).
- ▶ You should define `NMATMAX`, i.e. the maximum matrixsize (number of basis functions). This value should be adjusted according to the memory of your hardware. Rough estimates are:  
`NMATMAX=10000 ==> 1 GB (real, i.e. with inversion symmetry)`  
`NMATMAX=30000 ==> 10 GB (real) (==> cells with about 80-150 atoms/unitcell)`  
 If you choose it too large, `lapw1` will start to “page” leading to unacceptable performance or a crash. Please note: when running N k-point parallel jobs on a single PC, you need N-times this memory. In addition, `NMATMAX` will be automatically reduced (by  $\sqrt{2}$ ) for complex (without inversion) cases and increased by  $\sqrt{NPE}$  for mpi-parallel cases, where NPE is the number of mpi-parallel cores.
- ▶ Then 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. Additionally, you can change all specified options (compiler, paths, libraries,...) manually - **siteconfig\_lapw** saves them in the following files:

- ▶ **WIEN2k\_VERSION**: Contains the version of your WIEN2k installation
- ▶ **WIEN2k\_COMPILER**: Contains the chosen FORTRAN- and C-compiler, and, optionally, a MPI-compiler.
- ▶ **WIEN2k\_MPI**: Contains only one single keyword (MPI), if you specified that you want to run parallel calculations and directs siteconfig to compile (create) the mpi-versions too.
- ▶ **WIEN2k\_OPTIONS**: Contains all options needed to compile the programs of WIEN2k (paths to libraries, library names, preprocessor switches,...).
- ▶ **WIEN2k\_parallel\_options**: Contains all options needed for parallelization
- ▶ **WIEN2k\_SYSTEM**: Contains the specified system you run your calculations on.
- ▶ **WIEN2k\_INSTALLDATE**: Contains the timestamp of your installation of WIEN2k (the first time you ran **siteconfig\_lapw**. If you remove this file, siteconfig will fall back into the “first-time” installation mode and restart with siteconfigs defaults (ignoring previous settings).

### 11.2.4 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
- ▶ defines environment variables (**\$WIENROOT**, **\$SCRATCH**, **\$OMP\_NUM\_THREADS**)

to your `~/ .cshrc` or `~/ .bashrc` file.

We recommend in particular to set:

**\$OMP\_NUM\_THREADS** to 2 or 4 (assuming you have a 4 core processor). Don't set it larger than 8, even when you have many more shared memory cores;

**\$SCRATCH** to a local directory (and not your NFS-mounted home-directory) to reduce network traffic.

Optionally you may need to edit these files and set the **\$LD\_LIBRARY\_PATH** variable (path where compiler-libs or blas-libraries are located).

*Note: This will work only when the csh, tcsh or bash-shell is your login shell. Depending on your settings you may have to add similar lines also in your .login file. If you are using a different login-shell, edit your startup files manually.*

### 11.2.5 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 11.2.3).

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`; `lapw1_mpi`, `lapw1c_mpi`, `lapw2_mpi`, `lapw2c_mpi` are created using `make para` (`lapw0`) and `make rp`; `make cp`. `make all` produces all available executables.

Most of the CPU time will be spent in `lapw1` and (to a smaller extent) in `lapw2` and `lapw0`, except for hybrid-DFT calculations, where 90 % of the time goes into the `hf` program. 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 (like `AVX512` or `-qarch=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, aclm for AMD, essl for IBM) or use the optimized Linux `OpenBlas` ). Because of the superior performance of the Intel-mkl library we recommend `ifort/mkl` instead of `gfortan` (or some other commercial f90 compiler). If such libraries are not available use the `OpenBLAS`-library (<http://www.openblas.net/>), which is available on modern Linux versions.

### 11.2.6 Global dimensioning parameters

`WIEN2k` is written in Fortran 90 and all important arrays are allocated dynamically. The only important parameters left are `NMATMAX` and `NUME`, specifying the maximum matrixsize (should be adjusted to the memory of your hardware, see above) and the maximum number of eigenvalues (must be increased for unitcells with large number of electrons)

Some less important parameters are still present and described in chapter “dimensioning parameters” of the respective section in chapter 6.

We recommend to use `siteconfig_lapw` for redimensioning and recompilation. In order to work properly, the parameter `XXXX` in the respective `param.inc` files must obey the following syntax:

```
PARAMETER (XXXX= . . . .)
```

*Note: between “(”, `XXXX` and “=” there must be no space.*

## 11.3 Installation and Configuration of `w2web`

### 11.3.1 General issues

`w2web` requires `perl`, which should be available on most systems. (If not contact your system administrator or install it yourself from the WWW)

When you start `w2web` for the first time on the computer where you want to execute `WIEN2k` (you may have to telnet, ssh,.. to this machine) with the command `w2web [-p xxxx]`, you will be asked for a username/password (I recommend you use the same as for your UNIX login).

You must also specify a “port” number (which can be changed the next time you start `w2web`). If the default port (7890) used to serve the interface is already in use by some other process, you will get the error message `w2web failed to bind port 7890 - port already in use!`. Then you will have to choose a different port number (between 1024 and 65536) . Please remember this port number, you need it when connecting to the `w2web` server.

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

Once **w2web** has been started, use your favorite WWW-browser to connect to **w2web**, specifying the correct portnumber, e.g. **firefox http://hostname\_where\_w2web\_runs:7890**

On certain sites a firewall may block all high ports and one cannot connect to this machine. In these cases you can create a *ssh-tunnel* using the following commands:

At your "local\_host" (the PC in front of you) connect to the "w2web\_host" (where you started **w2web**) using

```
ssh -fNL 2000:w2web_host:7890 user@w2web_host
```

On your local host use a web browser and connect with: **firefox http:127.0.0.1:2000**.

Using "*Configuration*" you can further tailor the behaviour according to your wishes. In particular you can define new "execution types" to adjust to your queuing system.

For example the line

```
batch=batch < %f
```

defines an execution type "batch" using the UNIX batch command. (**w2web** collects its commands in a temporary script and you can access it using %f).

If you run on a machine with a queuing system (like loadleveler, sun-grid-engine, or pbs) you may define an "execution type"

```
qsub=cat %f > w2web-job;qsub-wienjob_lapw
```

The following scripts may serve as templates: **qsub-wienjob\_lapw** in **\$WIENROOT** needs a master-job-template **qsub-job0\_lapw** and examples for loadleveler and SGE are provided in **\$WIENROOT** (you may need to adapt them ! Other examples you can find on our FAQ-page on the web). Of course, with some small modifications you can define several "execution types" with eg. different number of processors or mpi vs. k-point parallel runs,....

**w2web** saves several variables in startup files which are in the (`~/w2web`) directory.

### 11.3.2 How does **w2web** work?

**w2web** acts like a normal web-server - except that it runs on a "user level port" instead of the default http-port 80. It serves html-files and executes perl-scripts or executes system or user commands on the server host.

### 11.3.3 **w2web**-files in you home directory

**w2web** creates on the first start of **w2web** on host "hostname" the directory **.w2web/hostname** in your home directory with the following content:

- ▶ `.w2web/hostname/conf`
- ▶ `.w2web/hostname/logs`
- ▶ `.w2web/hostname/sessions`

### 11.3.4 The configuration file `conf/w2web.conf`

In this file various configuration parameters are stored by **w2web**. To restrict the access to certain IP addresses you can add lines like:

```
deny=*****
allow=128.130.134.* 128.130.142.10
```

### 11.3.5 The password file `conf/w2web.users`

This file is created during the first run of **w2web**.

If you remove this file, the next start of **w2web** will activate the installation procedure again.

### 11.3.6 Using the https-protocol with **w2web**

In order to use the https-protocol the perl-library `Net::SSLeay` in addition to the OpenSSL package must be installed on your system. Both are freely available.

Then you must include a line with `ssl=1` in `w2web.conf`.

If you run **w2web-server** in `ssl-mode` you need a site certificate for your server. You may use the supplied certificate in `$WIENROOT/SRC_w2web/bin/w2web.pem` (copy this file to your `conf`-directory and set the `keyfile=~/.w2web/<hostname>/conf/w2web.pem` line in your `w2web.conf`).

This certificate will not expire until 2015, but usually browsers will complain that they do not know the Certificate Authority who issued this certificate - if you don't like this message, you must buy a certificate from VeriSign, Thawte or a similar CA.

Of course you must connect to `https:` instead of `http:`, i.e. use:

```
firefox 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-file systems, 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 XCrystal.

**OMP\_NUM\_THREADS** [1—2—4] on multi-core machines for parallelization using OpenMP and in certain libraries (`mkl`).

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\_EXTRAFINE** 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\_ - machine.file\_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 srun only commensurate grids are supported at the moment (i.e., the number of processes of your grid is a multiple or - 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



---

## 12 Trouble shooting

---

In this chapter hints are given for solving some difficulties that have occurred frequently. This chapter is by no means complete and the authors would appreciate further suggestions which might be useful for other users. Beside the printed version of the users guide, an online pdf version is available using **help\_lapw**. You can search for a specific keyword (use  $\wedge$ 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/mailling\\_list/](http://www.wien2k.at/reg_user/mailling_list/)  
and subscribe. You will then automatically be added to the mailing list  
[wien@theochem.tuwien.ac.at](mailto: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 exist, your **case.struct** file may contain an error. A check for overlapping spheres is also included in **mixer** and **lapw1**.

**case.outputd**: Possible stops or warnings are:

“NO SYMMETRY OPERATION FOUND IN ROTDEF”: This indicates that in your **case.struct** file either the positions of equivalent atoms are not specified correctly (only positive coordinates allowed!!) or the symmetry operations are wrong.

**case.output1**: Possible stops or warnings are:

“NO ENERGY LIMITS FOUND IN SELECT”: This indicates that  $E_{top}$  or  $E_{bottom}$  could not be found for some  $u_l(r, E_l)$ . Check your input if it happens in the zeroth iteration. Later, (usually in the second to sixth iteration) it may indicate that in your SCF cycle something went wrong and you are using a crazy potential. Usually it means that mixing of the charge densities was diverging and large charge fluctuations occurred. Check previous charges for being physically reasonable (grep for labels **:NTOxx :CTOxx :DIS :NEC01**). Usually this happens when your input is not ok, or for very ill conditioned problems

(very rare), or more likely, when “Ghostbands” appeared (or some states were missing) because of bad energy parameters in `case.in1`. You will probably have to delete `case.broy*` and `case.scf`, rerun `x dstart` and then change some calculational parameters. These could be: fixing some energy parameter (modify both, `case.in1` and `case.in1_orig` or try the `-in1orig` switch if you have used `-in1new`); switch to a broadening method (TEMP with eg. 0.010 mRy); or increase the k-mesh (magnetic metals); or reduce the mixing parameter in `case.inm` slightly (eg. to 0.1). In very difficult (magnetic) cases a PRATT mixing with eg. 0.01 mixing might be helpful at the beginning of the scf cycle (but later switch to MSEC1 again) !

“STOP RDC 22”: This indicates that the overlap matrix is not positive definite. This usually happens if your `case.struct` file has some error in the structure or if you have an (almost) linear dependent basis, which can happen for large RKMAX values and/or if you are using very different (extremely small and large) sphere radii  $R_{MT}$ .

“X EIGENVALUES BELOW THE ENERGY `emin`”: This indicates that X eigenvalues were found below `emin`. `Emin` is set in `case.in1` (see sec. 7.6.3) 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 7.6.3).

**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 oscillations (see SELECT error above)

If the SCF cycle stops somewhere (especially in the first few iterations), it is quite possible, that the source of the error is actually in a previous part of the cycle or even in a previous (e.g. the zeroth) iteration. Check in the `case.scf` file previous charges, eigenvalues, ... whether they are **physically reasonable** (see SELECT error above).

## 12.1 Ghost bands

Approximate linear dependence of the basis set or the linearization of the energy dependence of the radial wave functions (see section 2.2) can lead to spurious eigenvalues, termed “ghost bands”.

The first case may occur in a system which has atoms with very different atomic sphere radii. Suppose you calculate a hydroxide with very short O-H bonds so that you select small  $R_{MT}$  radii for O and H such as e.g. 1.0 and 0.6 a.u., respectively. The cation may be large and thus you could choose a large  $R_{MT}$  of e.g. 2.4 a.u. However, this gives a four time larger effective RKmax for the cation than for H, (e.g. 16.0 when you select RKmax=4.0 in **case.in1**). This enormous difference in the convergence may lead to unphysical eigenvalues. In such cases choose lmax=12 in **case.in1** (in order to get a very good re-expansion of the plane waves) and reduce  $R_{MT}$  for the cation to e.g. 1.8 a.u.

The second case can occur when you don't use a proper set of local orbitals. In this situation the energy region of interest (valence bands) falls about midway between two states with different principle quantum numbers, but with the same l-value (for one atom).

Take for example Ti with its 3*p* states being occupied as (semi-core) states, while the 4*p* states remain mostly unoccupied. In the valence band region neither of those two states (Ti 3*p*, 4*p*) 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*<sub>2</sub> (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 2*s* states, which are ok, since there are four O's per unit cell, four states are found.

The occurrence of such unphysical (indeed, unchemical!) ghostbands is the first warning that something went wrong. A more definite warning comes upon running LAPW2, where the corresponding charge densities are calculated. If the contribution to the charge density from the energy derivative of the basis function [the  $B_{lm}$  coefficient in equ. 2.4,2.7] is significant (i.e. much more than 5 per cent) then a warning is issued in LAPW2.

In the present example it reads:

```
QTL-B VALUE .EQ. 40.35396 !!!!!
```

This message is found in both the **case.scf** file and in **case.output2**.

When such a message appears, one can also look at the partial charges (QTL), which are printed under these conditions to OUTPUT2, and always appear in the files **case.helpXXX**, etc., where the last digit refers to the atomic index.

In the file below, note the E(1) energy parameter as well as the 6 ghost band energies around -1.29.

```
----- top of file:tio2.scf -----
      ATOMIC SPHERE DEPENDENT PARAMETERS FOR ATOM  Titanium
      OVERALL ENERGY PARAMETER IS          .2000
      E ( 0)=          .2000
---->    E ( 1)=          .2000
      E ( 2)=          .2000   E(BOTTOM)=    -.140   E(TOP)= -200.000

      ATOMIC SPHERE DEPENDENT PARAMETERS FOR ATOM  Oxygen
      OVERALL ENERGY PARAMETER IS          .2000
      E ( 0)=    -.7100   E(BOTTOM)=    -2.090   E(TOP)=    .670

      K=    .00000    .00000    .00000          1
:RKM : MATRIX SIZE= 599  RKM= 6.99  WEIGHT= 8.00  PGR:
      EIGENVALUES ARE:
      -1.2970782  -1.2970782  -1.2948747  -1.2897193  -1.2897193
      -1.2882306  -1.2882306  -1.2882306  -1.2882306  -1.2882306
      -1.2882306  -1.2882306  -1.2882306  -1.2882306  -1.2882306
      .1914068    .1914068    .1914068    .1914068    .1914068
      .3477629    .3477629    .3477629    .3477629    .3477629
      .3477629    .3477629    .3477629    .3477629    .3477629
      .5617155    .5617155    .5617155    .5617155    .5617155
      .8736991    .8736991    .8736991    .8736991    .8736991
      1.4285169
      *****
      NUMBER OF K-POINTS:          1
```

```

:NOE  : NUMBER OF ELECTRONS          = 48.000
:FER  : F E R M I - ENERGY          = .53562

:POS01: AT.NR. -1 POSITION = .00000 .00000 .00000 MULTIPLICITY= 2
      LMMAX=10
      LM= 0 0 2 0 2 2 4 0 4 2 4 4 6 0 6 2 6 4 6 6 0 0 0 0 0 0 0 0 0 0 0 0
:CHA01: TOTAL CHARGE INSIDE SPHERE 1 = 8.802166
:PCS01: PARTIAL CHARGES SPHERE = 1 S,P,D,F,PX,PY,PZ,D-Z2,D-X2Y2,D-XY,D-XZ,D-YZ
:QTL01: .127 6.080 2.518 .067 2.011 2.047 2.022 1.090 .760 .155 .480 .034
      VXX VYY VZZ UP TO R
:VZZ01: -4.96856 8.48379 -3.51524 2.000
:POS02: AT.NR. -2 POSITION = .30500 .30500 .00000 MULTIPLICITY= 4
      LMMAX=16
      LM= 0 0 1 0 2 0 2 2 3 0 3 2 4 0 4 2 4 4 5 0 5 2 5 4 6 0 6 2 6 4 6 6 0 0
:CHA02: TOTAL CHARGE INSIDE SPHERE 2 = 5.486185
:PCS02: PARTIAL CHARGES SPHERE = 2 S,P,D,F,PX,PY,PZ,D-Z2,D-X2Y2,D-XY,D-XZ,D-YZ
:QTL02: 1.559 3.902 .022 .002 1.296 1.306 1.300 .014 .004 .000 .003 .001
      VXX VYY VZZ UP TO R
:VZZ02: .25199 -.55091 .29892 1.600

:CHA : TOTAL CHARGE INSIDE CELL = 48.000000
:SUM : SUM OF EIGENVALUES = -15.810906

      QTL-B VALUE .EQ. 40.35396 !!!!!!!
      NBAND in QTL-file: 24
-----end of truncated file tio2.scf-----

```

Next we show **tio2.output2** for the first of the ghost bands at -1.297 Ry. One sees that it corresponds mainly to a p-like charge, which originates from the energy derivative part Q(UE) of the Kohn-Sham orbital. Q(UE) contributes 40.1% compared with 8.5% from the main component Q(U). Q(UE) greater than Q(U) is a good indication for a ghost band.

```

-----part of file tio2.output2 -----
      QTL-B VALUE .EQ. 40.35396 !!!!!!!
      K-POINT: .0000 .0000 .0000 599 36 1
      BAND # 1 E= -1.29708 WEIGHT= 2.0000000
      L= 0 L= 1 PX: PY: PZ: L= 2 DZ2: DX2Y2: DXY: DXZ: DYZ: L= 3
      QINSID: .0000 48.6035 35.0996 13.5039 .0000 .0000 .0000 .0000 .0000 .0000 .0000 .0030
      Q(U) : .0000 8.4902 6.0125 2.4777 .0000 .0000 .0000 .0000 .0000 .0000 .0000 .0026
      Q(UE) : .0000 40.1132 29.0871 11.0261 .0000 .0000 .0000 .0000 .0000 .0000 .0000 .0005
      L= 0 L= 1 PX: PY: PZ: L= 2 DZ2: DX2Y2: DXY: DXZ: DYZ: L= 3
      QINSID: .1294 .0707 .0000 .0055 .0653 .0088 .0038 .0049 .0000 .0000 .0000 .0022
      Q(U) : .1279 .0627 .0000 .0052 .0575 .0087 .0038 .0049 .0000 .0000 .0000 .0020
      Q(UE) : .0016 .0081 .0000 .0003 .0077 .0001 .0000 .0000 .0000 .0000 .0000 .0002
      QOUT : 1.9265
-----bottom of truncated file -----

```

Another file in which the same information can be found is **tio2.help031**, since the ghost band is caused by a bad choice for the Ti-p energy parameter:

```

-----Top of file tio2.help031 -----
      K-POINT: .0000 .0000 .0000 599 36 1
      BAND # 1 E= -1.29708 WEIGHT= 2.0000000
      L= 0 .00000 .00000 .00000 .00000 .00000 .00000
      L= 1 48.60346 8.49022 40.11324 .00000 .00000 .00000
      PX: 35.09960 6.01247 29.08712 .00000 .00000 .00000
      PY: 13.50386 2.47774 11.02612 .00000 .00000 .00000
      PZ: .00000 .00000 .00000 .00000 .00000 .00000
      L= 2 .00000 .00000 .00000 .00000 .00000 .00000
      DZ2: .00000 .00000 .00000 .00000 .00000 .00000
      DX2Y2: .00000 .00000 .00000 .00000 .00000 .00000
      DXY: .00000 .00000 .00000 .00000 .00000 .00000
      DXZ: .00000 .00000 .00000 .00000 .00000 .00000
      DYZ: .00000 .00000 .00000 .00000 .00000 .00000
      L= 3 .00304 .00255 .00050 .00000 .00000 .00000
      L= 4 .00000 .00000 .00000 .00000 .00000 .00000
      L= 5 .00096 .00082 .00014 .00000 .00000 .00000
      L= 6 .00000 .00000 .00000 .00000 .00000 .00000
-----bottom of truncated file -----

```

Note again for L=1 the percentage of charge associated with the primary (APW) basis functions ul (8.5%) versus that coming from the energy derivative component (40.1%).

If a ghost band appears, one should first analyze its origin as indicated above, then use appropriate local orbitals to improve the calculation and get rid of these unphysical states.

**Do not perform calculations with “ghost-bands”, even when the calculation converges.**

Good luck !





## **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 7.57 (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  $45^\circ$  (e.g. in the xy-plane)
- ▶ rotation of z into the (111) direction

Inspection of the output of SYMMETRY tells you if the local rotation matrix is the unit matrix or it gives you a clear indication how to find the proper matrix.

The local rotation matrix  $\mathbf{R}$ , which transforms the global coordinates  $r$  to the rotated ones  $r'$ , is defined by  $\mathbf{R}r = r'$ .

There are two simple ways to check the local rotation matrixes together with the selected LM combinations:

- ▶ charge density plots generated with LAPW5 must be continuous across the atomic sphere boundary (especially valence or difference density plots are very sensitive, see 8.14)
- ▶ 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 TiO<sub>2</sub>, which has already been described as an example in section 10.3. Also two other examples will be given (see below).

## A.1 Rutile ( $TiO_2$ )

Examine the output from symmetry. It should be obvious that you need local rotation matrices for both, Ti and O:

```

.....
Titanium operation # 1 1
Titanium operation # 2 -1
Titanium operation # 5 2 || z
Titanium operation # 6 m n z
Titanium operation # 12 m n 110
Titanium operation # 13 m n -110
Titanium operation # 18 2 || 110
Titanium operation # 19 2 || -110
pointgroup is mmm (neg. iatnr!!)
axes should be: m n z, m n y, m n x

```

This output tells you, that for Ti a mirror plan normal to z is present, but the mirror planes normal to x and y are missing. Instead, they are normal to the (110) plane and thus you need to rotate x, y by  $45^\circ$  around the z axis. (The required choice of the coordinate system for mmm symmetry is also given in Table 7.57)

```

.....
Oxygen operation # 1 1
Oxygen operation # 6 m n z
Oxygen operation # 13 m n -110
Oxygen operation # 18 2 || 110
pointgroup is mm2 (neg. iatnr!!)
axes should be: 2 || z, m n y

```

For O the 2-fold symmetry axes points into the (110) direction instead of z. The appropriate rotation matrices for Ti and O are:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \end{pmatrix}$$

## A.2 Si $\Gamma$ -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:

```

Si operation # 1 1
Si operation # 13 m n -110
Si operation # 16 m n -101
Si operation # 17 m n 0-11
Si operation # 24 3 || 111
Si operation # 38 3 || 111
pointgroup is 3m (neg. iatnr!!)
axis should be: 3 || z, m n y
lm: 0 0 1 0 2 0 3 0 3 3 4 0 4 3 5 0 5 3 6 0 6 3 6

```

Therefore the required local rotation matrix should rotate z into the (111) direction and thus the matrix in the "struct" file should be:

$$\begin{array}{cccc} 0.4082483 & -.7071068 & 0.5773503 & \frac{\sqrt{6}}{6} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ 0.4082483 & 0.7071068 & 0.5773503 & \frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ -.8164966 & 0.0000000 & 0.5773503 & -2\frac{\sqrt{6}}{6} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \end{array}$$

### A.3 Trigonal Selenium

Selenium possesses space group P3121 with the following struct file:

```
H      LATTICE,NONEQUIV.ATOMS: 1
MODE OF CALC=RELA  POINTGROUP:32
8.2500000 8.2500000 9.369000
ATOM= -1: X= .7746000  Y= .7746000  Z= 0.0000000
      MULT= 3          ISPLIT= 8
ATOM= -1: X= .2254000  Y= .0000000  Z= 0.3333333
ATOM= -1: X= .0000000  Y= .2254000  Z= 0.6666667
Se      NPT= 381  R0=.000100000  RMT=2.100000000  Z:34.0
LOCAL ROT.MATRIX:  0.0      0.5000000  0.8660254
                   0.0000000  -.8660254  0.5000000
                   1.0000000  0.0000000  0.0
      6      IORD OF GROUP G0
.....
```

The output of SYMMETRY reads:

```
Se      operation # 1      1
Se      operation # 9      2  $|$$|$ 110
      pointgroup is 2 (neg. iatnr!!)
      axis should be: 2 || z
lm: 0 0 1 0 2 0 2 2 -2 2 3 0 3 2 -3 2 4 0 4 2 -4 2 .....
```

Point group 2 should have its 2-fold rotation axis along z, so the local rotation matrix can be constructed in two steps: firstly interchange x and z (that leads to z || (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

<b>1H</b> Hs																	<b>2He</b> He <sup>2</sup>
<b>3Li</b> He2s <sup>1</sup>	<b>4Be</b> He2s <sup>2</sup>															<b>9F</b> He2s <sup>2</sup> p <sup>5</sup>	<b>10Ne</b> He2s <sup>2</sup> p <sup>6</sup>
<b>11Na</b> Ne3s <sup>1</sup>	<b>12Mg</b> Ne3s <sup>2</sup>															<b>17Cl</b> Ne3s <sup>2</sup> p <sup>5</sup>	<b>18Ar</b> Ne3s <sup>2</sup> p <sup>6</sup>
<b>19K</b> Ar4s <sup>1</sup>	<b>20Ca</b> Ar4s <sup>2</sup>	<b>21Sc</b> Ar3d <sup>1</sup> 4s <sup>2</sup>	<b>22Ti</b> Ar3d <sup>2</sup> 4s <sup>2</sup>	<b>23V</b> Ar3d <sup>3</sup> 4s <sup>2</sup>	<b>24Cr</b> Ar3d <sup>5</sup> 4s <sup>1</sup>	<b>25Mn</b> Ar3d <sup>5</sup> 4s <sup>2</sup>	<b>26Fe</b> Ar3d <sup>6</sup> 4s <sup>2</sup>	<b>27Co</b> Ar3d <sup>7</sup> 4s <sup>2</sup>	<b>28Ni</b> Ar3d <sup>8</sup> 4s <sup>2</sup>	<b>29Cu</b> Ar3d <sup>10</sup> 4s <sup>1</sup>	<b>30Zn</b> Ar3d <sup>10</sup> 4s <sup>2</sup>	<b>31Ga</b> Ar3d <sup>10</sup> 4s <sup>2</sup> 4p <sup>1</sup>	<b>32Ge</b> Ar3d <sup>10</sup> 4s <sup>2</sup> 4p <sup>2</sup>	<b>33As</b> Ar3d <sup>10</sup> 4s <sup>2</sup> 4p <sup>3</sup>	<b>34Se</b> Ar3d <sup>10</sup> 4s <sup>2</sup> 4p <sup>4</sup>	<b>35Br</b> Ar3d <sup>10</sup> 4s <sup>2</sup> 4p <sup>5</sup>	<b>36Kr</b> Ar3d <sup>10</sup> 4s <sup>2</sup> 4p <sup>6</sup>
<b>37Rb</b> Kr5s <sup>1</sup>	<b>38Sr</b> Kr5s <sup>2</sup>	<b>39Y</b> Kr4d <sup>1</sup> 5s <sup>2</sup>	<b>40Zr</b> Kr4d <sup>2</sup> 5s <sup>2</sup>	<b>41Nb</b> Kr4d <sup>4</sup> 5s <sup>1</sup>	<b>42Mo</b> Kr4d <sup>5</sup> 5s <sup>1</sup>	<b>43Tc</b> Kr4d <sup>5</sup> 5s <sup>2</sup>	<b>44Ru</b> Kr4d <sup>7</sup> 5s <sup>1</sup>	<b>45Rh</b> Kr4d <sup>8</sup> 5s <sup>1</sup>	<b>46Pd</b> Kr4d <sup>10</sup> 5s <sup>0</sup>	<b>47Ag</b> Kr4d <sup>10</sup> 5s <sup>1</sup>	<b>48Cd</b> Kr4d <sup>10</sup> 5s <sup>2</sup>	<b>49In</b> Kr4d <sup>10</sup> 5s <sup>2</sup> 5p <sup>1</sup>	<b>50Sn</b> Kr4d <sup>10</sup> 5s <sup>2</sup> 5p <sup>2</sup>	<b>51Sb</b> Kr4d <sup>10</sup> 5s <sup>2</sup> 5p <sup>3</sup>	<b>52Te</b> Kr4d <sup>10</sup> 5s <sup>2</sup> 5p <sup>4</sup>	<b>53I</b> Kr4d <sup>10</sup> 5s <sup>2</sup> 5p <sup>5</sup>	<b>54Xe</b> Kr4d <sup>10</sup> 5s <sup>2</sup> 5p <sup>6</sup>
<b>55Cs</b> Xe6s <sup>1</sup>	<b>56Ba</b> Xe6s <sup>2</sup>	<b>57-71</b> <b>La-Lu</b>	<b>72Hf</b> Xe4f <sup>14</sup> 5d <sup>2</sup> 6s <sup>2</sup>	<b>73Ta</b> Xe4f <sup>14</sup> 5d <sup>3</sup> 6s <sup>2</sup>	<b>74W</b> Xe4f <sup>14</sup> 5d <sup>4</sup> 6s <sup>2</sup>	<b>75Re</b> Xe4f <sup>14</sup> 5d <sup>5</sup> 6s <sup>2</sup>	<b>76Os</b> Xe4f <sup>14</sup> 5d <sup>6</sup> 6s <sup>2</sup>	<b>77Ir</b> Xe4f <sup>14</sup> 5d <sup>7</sup> 6s <sup>2</sup>	<b>78Pt</b> Xe4f <sup>14</sup> 5d <sup>9</sup> 6s <sup>1</sup>	<b>79Au</b> Xe4f <sup>14</sup> 5d <sup>10</sup> 6s <sup>1</sup>	<b>80Hg</b> Xe4f <sup>14</sup> 5d <sup>10</sup> 6s <sup>2</sup>	<b>81Tl</b> Xe4f <sup>14</sup> 5d <sup>10</sup> 6s <sup>2</sup> 6p <sup>1</sup>	<b>82Pb</b> Xe4f <sup>14</sup> 5d <sup>10</sup> 6s <sup>2</sup> 6p <sup>2</sup>	<b>83Bi</b> Xe4f <sup>14</sup> 5d <sup>10</sup> 6s <sup>2</sup> 6p <sup>3</sup>	<b>84Po</b> Xe4f <sup>14</sup> 5d <sup>10</sup> 6s <sup>2</sup> 6p <sup>4</sup>	<b>85At</b> Xe4f <sup>14</sup> 5d <sup>10</sup> 6s <sup>2</sup> 6p <sup>5</sup>	<b>86Rn</b> Xe4f <sup>14</sup> 5d <sup>10</sup> 6s <sup>2</sup> 6p <sup>6</sup>
<b>87Fr</b> Rn7s <sup>1</sup>	<b>88Ra</b> Rn7s <sup>2</sup>	89-103 <b>Ac-Lr</b>															
<b>57La</b> Xe5d <sup>1</sup> 6s <sup>2</sup>	<b>58Ce</b> Xe4f <sup>1</sup> 6s <sup>2</sup>	<b>59Pr</b> Xe4f <sup>3</sup> 6s <sup>2</sup>	<b>60Nd</b> Xe4f <sup>4</sup> 6s <sup>2</sup>	<b>61Pm</b> Xe4f <sup>5</sup> 6s <sup>2</sup>	<b>62Sm</b> Xe4f <sup>6</sup> 6s <sup>2</sup>	<b>63Eu</b> Xe4f <sup>7</sup> 6s <sup>2</sup>	<b>64Gd</b> Xe4f <sup>7</sup> 5d <sup>1</sup> 6s <sup>2</sup>	<b>65Tb</b> Xe4f <sup>9</sup> 6s <sup>2</sup>	<b>66Dy</b> Xe4f <sup>10</sup> 6s <sup>2</sup>	<b>67Ho</b> Xe4f <sup>11</sup> 6s <sup>2</sup>	<b>68Er</b> Xe4f <sup>12</sup> 6s <sup>2</sup>	<b>69Tm</b> Xe4f <sup>13</sup> 6s <sup>2</sup>	<b>70Yb</b> Xe4f <sup>14</sup> 6s <sup>2</sup>	<b>71Lu</b> Xe4f <sup>14</sup> 5d <sup>1</sup> 6s <sup>2</sup>			
<b>89Ac</b> Rn6d <sup>1</sup> 7s <sup>2</sup>	<b>90Th</b> Rn6f <sup>14</sup> 7s <sup>2</sup>	<b>91Pa</b> Rn5f <sup>2</sup> 6d <sup>1</sup> 7s <sup>2</sup>	<b>92U</b> Rn5f <sup>3</sup> 6d <sup>1</sup> 7s <sup>2</sup>	<b>93Np</b> Rn5f <sup>4</sup> 6d <sup>1</sup> 7s <sup>2</sup>	<b>94Pu</b> Rn5f <sup>6</sup> 7s <sup>2</sup>	<b>95Am</b> Rn5f <sup>7</sup> 7s <sup>2</sup>	<b>96Cm</b> Rn5f <sup>7</sup> 6d <sup>1</sup> 7s <sup>2</sup>	<b>97Bk</b> Rn5f <sup>7</sup> 7s <sup>2</sup>	<b>98Cf</b> Rn5f <sup>10</sup> 7s <sup>2</sup>	<b>99Es</b> Rn5f <sup>11</sup> 7s <sup>2</sup>	<b>100Fm</b> Rn5f <sup>12</sup> 7s <sup>2</sup>	<b>101Md</b> Rn5f <sup>13</sup> 7s <sup>2</sup>	<b>102No</b> Rn5f <sup>14</sup> 7s <sup>2</sup>	<b>103Lr</b> Rn5f <sup>14</sup> 6d <sup>1</sup> 7s <sup>2</sup>			





**WIEN2k**

ISBN 3-9501031-1-2

---

# Bibliography

---

- [Tab, 1964] (1964). *International Tables for X-Ray Crystallography*, volume I. Kynoch, Birmingham. 44, 45, 46
- [Abt et al., 1994] Abt, A., Ambrosch-Draxl, C., and Knoll, P. (1994). *Physica B*, 194-196:1451. 11, 208
- [Abt, 1997] Abt, R. (1997). PhD Thesis, University of Graz. 11
- [Ahmed et al., 2013] Ahmed, S., Kivinen, J., Zaporzan, B., Curiel, L., Pichardo, S., and Rubel, O. (2013). BerryPi: A software for studying polarization of crystalline solids with wien2k density functional all-electron package. *Computer Physics Communications*, 184:647. 114
- [Ambrosch-Draxl et al., 1995] Ambrosch-Draxl, C., Majewski, J. A., Vogl, P., and Leising, G. (1995). *Phys. Rev. B*, 51:9668. 11
- [Ambrosch-Draxl and Sofo, 2006] Ambrosch-Draxl, C. and Sofo, J. (2006). *Comput. Phys. Commun.*, 175:1. 11, 122, 208
- [Andersen, 1973] Andersen, O. K. (1973). *Solid State Commun.*, 13:133. 8
- [Andersen, 1975] Andersen, O. K. (1975). *Phys. Rev. B*, 12:3060. 8
- [Anisimov et al., 1993] Anisimov, V. I., Solovyev, I. V., Korotin, M. A., Czyżyk, M. T., and Sawatzky, G. A. (1993). *Phys. Rev. B*, 48:16929. 147
- [Anisimov et al., 1991] Anisimov, V. I., Zaanen, J., and Andersen, O. K. (1991). *Phys. Rev. B*, 44:943. 147
- [Armiento and Kümmel, 2013] Armiento, R. and Kümmel, S. (2013). *Phys. Rev. Lett.*, 111:036402. 138, 141
- [Armiento and Mattsson, 2005] Armiento, R. and Mattsson, A. E. (2005). *Phys. Rev. B*, 72:085108. 138, 139, 141, 142
- [Bader, ] Bader, R. F. W. [<http://www.chemistry.mcmaster.ca/bader/aim/>]. 174
- [Bagheri and Blaha, 2019] Bagheri, M. and Blaha, P. (2019). *J. Electron Spectrosc. Relat. Phenom.*, 230:1--9. 11, 212, 213
- [Bartók and Yates, 2019] Bartók, A. P. and Yates, J. R. (2019). *J. Chem. Phys.*, 150:161101. 138, 139
- [Becke, 1988] Becke, A. D. (1988). *Phys. Rev. A*, 38:3098. 138, 141
- [Becke, 1993] Becke, A. D. (1993). *J. Chem. Phys.*, 98:5648. 138, 141

- [Becke and Edgecombe, 1990] Becke, A. D. and Edgecombe, K. E. (1990). *J. Chem. Phys.*, 92:5397. 141
- [Becke and Johnson, 2006] Becke, A. D. and Johnson, E. R. (2006). *J. Chem. Phys.*, 124:221101. 61, 141
- [Becke and Roussel, 1989] Becke, A. D. and Roussel, M. R. (1989). *Phys. Rev. A*, 39:3761. 141
- [Belbase et al., 2021] Belbase, K., Tröster, A., and Blaha, P. (2021). Stress tensor in the linearized augmented plane wave method. *Physical Review B*, 104:174113. 11, 89
- [Berland and Hyldgaard, 2014] Berland, K. and Hyldgaard, P. (2014). *Phys. Rev. B*, 89:035412. 68
- [Betzinger et al., 2010] Betzinger, M., Friedrich, C., and Blügel, S. (2010). *Phys. Rev. B*, 81:195117. 62, 78
- [Blaha et al., 2009] Blaha, P., Hofstätter, H., Koch, R., Laskowski, R., and Schwarz, K. (2009). *J. Comput. Phys.*, 229:453. 151
- [Blaha and Schwarz, 1983] Blaha, P. and Schwarz, K. (1983). *Int. J. Quantum Chem.*, 23:1535. 8
- [Blaha et al., 1985] Blaha, P., Schwarz, K., and Herzig, P. (1985). *Phys. Rev. Lett.*, 54:1192. 8
- [Blöchl et al., 1994] Blöchl, P. E., Jepsen, O., and Andersen, O. K. (1994). *Phys. Rev. B*, 49:16223. 11, 130, 163, 226
- [Boese et al., 2000] Boese, A. D., Doltsinis, N. L., Handy, N. C., and Sprik, M. (2000). *J. Chem. Phys.*, 112:1670. 138, 139, 141, 142
- [Boese and Handy, 2001] Boese, A. D. and Handy, N. C. (2001). *J. Chem. Phys.*, 114:5497. 138, 139, 141, 142
- [Brandenburg et al., 2016] Brandenburg, J.G. Bates, J., Sun, J., and Perdew, J. P. (2016). *Phys. Rev. B*, 94:115144. 144
- [Brooks, 1985] Brooks, M. S. S. (1985). *Physica B*, 130:6. 147
- [Burke et al., 2014] Burke, K., Cancio, A., Gould, T., and Pittalis, S. (2014). *arXiv:1409.4834*. 139, 142
- [Caldeweyher et al., 2017] Caldeweyher, E., Bannwarth, C., and Grimme, S. (2017). *J. Chem. Phys.*, 147:034112. 8, 66, 145
- [Caldeweyher et al., 2019] Caldeweyher, E., Ehlert, S., Hansen, A., Neugebauer, H., Spicher, S., Bannwarth, C., and Grimme, S. (2019). *J. Chem. Phys.*, 150:154122. 8, 66, 145
- [Caldeweyher et al., 2020] Caldeweyher, E., Mewes, J.-M., Ehlert, S., and Grimme, S. (2020). *Phys. Chem. Chem. Phys.*, 22:8499. 8, 66, 145
- [Cancio et al., 2018] Cancio, A., Chen, G. P., Krull, B. T., and Burke, K. (2018). *J. Chem. Phys.*, 149:084116. 139, 142
- [Carmona-Espíndola et al., 2015] Carmona-Espíndola, J., Gázquez, J. L., Vela, A., and Trickey, S. B. (2015). *J. Chem. Phys.*, 142:054105. 138, 141

- [Chan et al., 2009] Chan, T.-L., Wang, C. Z., Ho, K. M., and Chelikowsky, J. R. (2009). *Phys. Rev. Lett.*, 102:176101. 174
- [Charpin, 2001] Charpin, T. (2001). see \$WIENROOT/SRC/elast-UG.ps. 182
- [Constantin et al., 2016] Constantin, L. A., Terentjevs, A., Della Sala, F., Cortona, P., and Fabiano, E. (2016). *Phys. Rev. B*, 93:045126. 138, 139, 141, 142
- [Cooper, 2010] Cooper, V. R. (2010). *Phys. Rev. B*, 81:161104(R). 68
- [Czyzyk and Sawatzky, 1994] Czyzyk, M. T. and Sawatzky, G. A. (1994). *Phys. Rev. B*, 49:14211. 147
- [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, 127, 167
- [Desclaux, 1975] Desclaux, J. P. (1975). *Comput. Phys. Commun.*, 9:31. This paper contains much of the Dirac-Fock treatment used in Desclaux's relativistic LSDA code. 127, 167
- [Dion et al., 2004] Dion, M., Rydberg, H., Schröder, E., Langreth, D. C., and Lundqvist, B. I. (2004). *Phys. Rev. Lett.*, 92:246401. 8, 66, 67, 146
- [Doumont et al., 2019] Doumont, J., Tran, F., and Blaha, P. (2019). *Phys. Rev. B*, 99:115101. 65
- [Doumont et al., 2022] Doumont, J., Tran, F., and Blaha, P. (2022). Implementation of self-consistent mgga functionals in augmented plane wave based methods. *Phys. Rev. B*, 105:195138. 68, 69
- [Engel and Vosko, 1993] Engel, E. and Vosko, S. H. (1993). *Phys. Rev. B*, 47:13164. 138, 141
- [Eriksson and Johansson, 1989] Eriksson, O. and Johansson, B. (1989). *J. Phys.: Condens. Matter*, 1:4005. 147
- [Fabiano et al., 2010] Fabiano, E., Constantin, L. A., and Della Sala, F. (2010). *Phys. Rev. B*, 82:113104. 138, 139, 141, 142
- [Ferreira et al., 2008] Ferreira, L. G., Marques, M., and Teles, L. K. (2008). *Phys. Rev. B*, 78:125116. 64, 65
- [Furness et al., 2020] Furness, J. W., Kaplan, A. D., Ning, J., Perdew, J. P., and Sun, J. (2020). *J. Phys. Chem. Lett.*, 11:8208. 138, 139
- [Gay, 1983] Gay, D. M. (1983). *ACM Trans. Math. Software*, 9:503. 204
- [Grimme et al., 2010] Grimme, S., Antony, J., Ehrlich, S., and Krieg, H. (2010). *J. Chem. Phys.*, 132:154104. 8, 66, 144
- [Grimme et al., 2011] Grimme, S., Ehrlich, S., and Goerigk, L. (2011). Effect of the damping function in dispersion corrected density functional theory. *J. Comput. Chem.*, 32:1456. 8, 66, 144

- [Gritsenko et al., 1995] Gritsenko, O., van Leeuwen, R., van Lenthe, E., and Baerends, E. J. (1995). *Phys. Rev. A*, 51:1944. 64
- [Haas et al., 2010] Haas, P., Tran, F., Blaha, P., Pedroza, L. S., da Silva, A. J. R., Odashima, M. M., and Capelle, K. (2010). *Phys. Rev. B*, 81:125136. 138, 139, 141, 142
- [Haas et al., 2011] Haas, P., Tran, F., Blaha, P., and Schwarz, K. (2011). *Phys. Rev. B*, 83:205117. 138, 141
- [Hamada, 2014] Hamada, I. (2014). *Phys. Rev. B*, 89:121103(R). 68
- [Hammer et al., 1999] Hammer, B., Hansen, L. B., and Nørskov, J. K. (1999). *Phys. Rev. B*, 59:7413. 138, 141
- [Hamprecht et al., 1998] Hamprecht, F. A., Cohen, A. J., Tozer, D. J., and Handy, N. C. (1998). *J. Chem. Phys.*, 109:6264. 138, 139, 141, 142
- [Henderson et al., 2008] Henderson, T. M., Janesko, B. G., and Scuseria, G. E. (2008). *J. Chem. Phys.*, 128:194105. 56, 138, 141
- [Heyd et al., 2003] Heyd, J., Scuseria, G. E., and Ernzerhof, M. (2003). *J. Chem. Phys.*, 118:8207. 56
- [Hirst, 1997] Hirst, L. L. (1997). *Rev. Mod. Phys.*, 69:607. 148
- [Hohenberg and Kohn, 1964] Hohenberg, P. and Kohn, W. (1964). *Phys. Rev.*, 136:B864. 7
- [Jamal et al., 2018] Jamal, M., Bilal, M., Ahmad, I., and Jalali-Asadabadi, S. (2018). *J. Alloys Compd.*, 735:569. 188
- [Jamal and Reshak, 2018] Jamal, M. and Reshak, A. (2018). *J. Phys.Chem.Solids*, 116:131. 92
- [Jansen and Freeman, 1984] Jansen, H. J. F. and Freeman, A. J. (1984). *Phys. Rev. B*, 30:561. 8
- [Kalantari et al., 2021] Kalantari, L., Tran, F., and Blaha, P. (2021). Density analysis for estimating the degree of on-site correlation on transition-metal atoms in extended systems. *Physical Review B*, 104:155127. 63
- [Kara and Kurki-Suonio, 1981] Kara, M. and Kurki-Suonio, K. (1981). *Acta Cryst.*, A37:201. xv, 164
- [Karsai et al., 2017] Karsai, F., Tran, F., and Blaha, P. (2017). *Comput. Phys. Commun.*, 220:230. 10, 151
- [King-Smith and Vanderbilt, 1993] King-Smith, R. D. and Vanderbilt, D. (1993). *Phys. Rev. B*, 47:1651. 114
- [Klimeš et al., 2010] Klimeš, J., Bowler, D. R., and Michaelides, A. (2010). *J. Phys.: Condens. Matter*, 22:022201. 68, 138, 141
- [Klimeš et al., 2011] Klimeš, J., Bowler, D. R., and Michaelides, A. (2011). *Phys. Rev. B*, 83:195131. 67, 68, 138, 141
- [Koelling, 1972] Koelling, D. D. (1972). *J. Phys. Chem. Solids*, 33:1335. 8

- [Koelling and Arberman, 1975] Koelling, D. D. and Arberman, G. O. (1975). *J. Phys. F: Met. Phys.*, 5:2041. 8, 151
- [Koelling and Harmon, 1977] Koelling, D. D. and Harmon, B. N. (1977). *J. Phys. C: Solid State Phys.*, 10:3107. 8
- [Kohler et al., 1996] Kohler, B., Wilke, S., Scheffler, M., Kouba, R., and Ambrosch-Draxl, C. (1996). *Comput. Phys. Commun.*, 94:31. 11
- [Kohn and Sham, 1965] Kohn, W. and Sham, L. J. (1965). *Phys. Rev.*, 140:A1133. 7, 138, 141
- [Kokalj, 2003] Kokalj, A. (2003). *Comput. Mater. Sci.*, 28:155. 247
- [Koller et al., 2012] Koller, D., Tran, F., and Blaha, P. (2012). *Phys. Rev. B*, 85:155109. 63
- [Kovacs et al., 2022] Kovacs, P., Tran, F., Blaha, P., and Madsen, G. K. (2022). What is the optimal mGGA exchange functional for solids? *J. Chemical Physics*, 157:094110. 86
- [Krieger et al., 1990] Krieger, J. B., Li, Y., and Iafate, G. J. (1990). *Phys. Lett. A*, 146:256. 61, 141
- [Krimmel et al., 1994] Krimmel, H. G., Ehmann, J., Elsässer, C., Fähnle, M., and Soler, J. M. (1994). *Phys. Rev. B*, 50:8846. 11
- [Krukau et al., 2006] Krukau, A. V., Vydrov, O. A., Izmaylov, A. F., and Scuseria, G. E. (2006). *J. Chem. Phys.*, 125:224106. 56
- [Kuisma et al., 2010] Kuisma, M., Ojanen, J., Enkovaara, J., and Rantala, T. T. (2010). *Phys. Rev. B*, 82:115106. 64, 141
- [Kuneš et al., 2001] Kuneš, J., Novák, P., Schmid, R., Blaha, P., and Schwarz, K. (2001). *Phys. Rev. B*, 64:153102. 11, 52, 158
- [Laskowski and Blaha, 2012a] Laskowski, R. and Blaha, P. (2012a). *Phys. Rev. B*, 85:035132. 106, 110, 207
- [Laskowski and Blaha, 2012b] Laskowski, R. and Blaha, P. (2012b). *Phys. Rev. B*, 85:245117. 106, 110, 207
- [Laskowski and Blaha, 2014] Laskowski, R. and Blaha, P. (2014). *Phys. Rev. B*, 89:014402. 106, 110, 207
- [Laskowski and Blaha, 2015a] Laskowski, R. and Blaha, P. (2015a). *Journal of Physical Chemistry C*, 119:731. 110
- [Laskowski and Blaha, 2015b] Laskowski, R. and Blaha, P. (2015b). *Journal of Physical Chemistry C*, 119:19390. 110
- [Laskowski et al., 2013] Laskowski, R., Blaha, P., and Tran, F. (2013). *Phys. Rev. B*, 87:195130. 207
- [Laskowski et al., 2017] Laskowski, R., Khoo, K., Haarmann, F., and Blaha, P. (2017). *Journal of Physical Chemistry C*, in print:0. 110, 111
- [Lee et al., 1988] Lee, C., Yang, W., and Parr, R. G. (1988). *Phys. Rev. B*, 37:785. 139, 142

- [Lee et al., 2010] Lee, K., Murray, E. D., Kong, L., Lundqvist, B. I., and Langreth, D. C. (2010). *Phys. Rev. B*, 82:081101(R). 68, 146
- [Lehtola et al., 2018] Lehtola, S., Steigemann, C., Oliveira, M. J. T., and Marques, M. A. L. (2018). *SoftwareX*, 7:1. 137, 261
- [Lieberman et al., 1965] Lieberman, D., Waber, J., and Cromer, D. (1965). Self-consistent-field dirac-slater wave functions for atoms and ions. i. comparison with previous calculations. *Physical Review A*, 137:27. 129
- [Liechtenstein et al., 1995] Liechtenstein, A. I., Anisimov, V. I., and Zaanen, J. (1995). *Phys. Rev. B*, 52:R5467. 147
- [MacDonald et al., 1980] MacDonald, A. H., Pickett, W. E., and Koelling, D. D. (1980). *J. Phys. C: Solid State Phys.*, 13:2675. 8, 11
- [Madsen, 2007] Madsen, G. K. H. (2007). *Phys. Rev. B*, 75:195108. 138, 141
- [Madsen et al., 2001] Madsen, G. K. H., Blaha, P., Schwarz, K., Sjöstedt, E., and Nordström, L. (2001). *Phys. Rev. B*, 64:195134. 10, 11, 151
- [Marks, 2013] Marks, L. D. (2013). *J. Chem. Theory Comput.*, 9:2786. 93, 169
- [Marks, 2021] Marks, L. D. (2021). Predictive mixing for density functional theory (and other fixed-point problems). *J. Chem. Theory Comput*, 17:5715. 93, 169
- [Marks and Luke, 2008] Marks, L. D. and Luke, R. (2008). *Phys. Rev. B*, 78:075114. 93, 169
- [Marques et al., 2012] Marques, M. A. L., Oliveira, M. J. T., and Burnus, T. (2012). *Comput. Phys. Commun.*, 183:2272. 137, 261
- [Mattheis and Hamann, 1986] Mattheis, L. F. and Hamann, D. R. (1986). *Phys. Rev. B*, 33:823. 8
- [Mejia-Rodriguez and Trickey, 2018] Mejia-Rodriguez, D. and Trickey, S. B. (2018). Deorbitalized meta-gga exchange-correlation functionals in solids. *Phys. Rev. B*, 98:115161. 138, 139
- [Mostofi et al., 2008] Mostofi, A. A., Yates, J. R., Lee, Y.-S., Souza, I., Vanderbilt, D., and Marzari, N. (2008). *Comput. Phys. Commun.*, 178:685. 112
- [Murnaghan, 1944] Murnaghan, F. D. (1944). *Proc. Natl. Acad. Sci. U.S.A.*, 30:244. 239
- [Neckel et al., 1975] Neckel, A., Rastl, P., Eibler, R., Weinberger, P., and Schwarz, K. (1975). *Microchim. Acta Suppl.*, 6:257. 11, 229
- [Novák, 1997] Novák, P. (1997). see \$WIENROOT/SRC/novak\_lecture\_on\_spinorbit.ps. 8, 11, 158
- [Novák, 2001] Novák, P. (2001). see \$WIENROOT/SRC/novak\_lecture\_on\_ldaumatricxelements.ps. 11



- [Novák, 2006] Novák, P. (2006). see \$WIENROOT/SRC/Bhf-3.ps. 166
- [Novák et al., 2001] Novák, P., Boucher, F., Gressier, P., Blaha, P., and Schwarz, K. (2001). *Phys. Rev. B*, 63:235114. 147
- [Ortenzi et al., 2012] Ortenzi, L., Mazin, I. I., Blaha, P., and Boeri, L. (2012). *Phys. Rev. B*, 86:064437. 137
- [Paier et al., 2006] Paier, J., Marsman, M., Hummer, K., Kresse, G., Gerber, I. C., and Ángyán, J. G. (2006). *J. Chem. Phys.*, 124:154709. 59
- [Pardini et al., 2012] Pardini, L., Bellini, V., Manghi, F., and Ambrosch-Draxl, C. (2012). *Comput. Phys. Commun.*, 183:628. 208
- [Peng and Perdew, 2017] Peng, H. and Perdew, J. P. (2017). *Phys. Rev. B*, 95:081105(R). 68
- [Peng et al., 2016] Peng, H., Yang, Z.-H., Perdew, J. P., and Sun, J. (2016). *Phys. Rev. X*, 6:041005. 68
- [Perdew et al., 1996] Perdew, J. P., Burke, K., and Ernzerhof, M. (1996). *Phys. Rev. Lett.*, 77:3865. 8, 128, 136, 138, 139, 141, 142
- [Perdew et al., 1992] Perdew, J. P., Chevary, J. A., Vosko, S. H., Jackson, K. A., Pederson, M. R., Singh, D. J., and Fiolhais, C. (1992). *Phys. Rev. B*, 46:6671. 8, 138, 139, 141, 142
- [Perdew et al., 1999] Perdew, J. P., Kurth, S., Zupan, A., and Blaha, P. (1999). *Phys. Rev. Lett.*, 82:2544. 8, 138, 139
- [Perdew et al., 2009] Perdew, J. P., Ruzsinszky, A., Csonka, G. I., Constantin, L. A., and Sun, J. (2009). *Phys. Rev. Lett.*, 103:026403. 138, 139
- [Perdew et al., 2008] Perdew, J. P., Ruzsinszky, A., Csonka, G. I., Vydrov, O. A., Scuseria, G. E., Constantin, L. A., Zhou, X., and Burke, K. (2008). *Phys. Rev. Lett.*, 100:136406. 128, 136, 138, 139, 141, 142
- [Perdew and Wang, 1992] Perdew, J. P. and Wang, Y. (1992). *Phys. Rev. B*, 45:13244. 7, 128, 136, 139, 142
- [Poirier and Tarantola, 1998] Poirier, J. and Tarantola, A. (1998). *Phys. Earth Planet. Inter.*, 109:1. 239
- [Rauch et al., 2020] Rauch, T., Marques, M. A. L., and Botti, S. (2020). *J. Chem. Theory Comput.*, 16:2654. 8, 63, 86, 136, 141
- [Reshak and Jamal, 2013] Reshak, A. and Jamal, M. (2013). *J. Alloys Compd.*, 555:362. 92
- [Resta et al., 1993] Resta, R., Posternak, M., and Baldereschi, A. (1993). *Phys. Rev. Lett.*, 70:1010. 114
- [Román-Pérez and Soler, 2009] Román-Pérez, G. and Soler, J. M. (2009). *Phys. Rev. Lett.*, 103:096102. 66, 146
- [Rubel et al., 2021] Rubel, O., Tran, F., Rocquefelte, X., and Blaha, P. (2021). Perturbation approach to ab initio effective mass calculations. *Computer Physics Communications*, 261:107648. 207

- [Ruzsinszky et al., 2009] Ruzsinszky, A., Csonka, G. I., and Scuseria, G. E. (2009). *J. Chem. Theory Comput.*, 5:763. 138, 139, 141, 142
- [Sabatini et al., 2013] Sabatini, R., Gorni, T., and de Gironcoli, S. (2013). *Phys. Rev. B*, 87:041108(R). 68, 146
- [Saini et al., 2022] Saini, H., Laurien, M., Blaha, P., and Rubel, O. (2022). Wloopphi: A tool for ab initio characterization of weyl semimetals. *Computer Physics Communications*, 270:108147. 117
- [Schwarz and Blaha, 1996] Schwarz, K. and Blaha, P. (1996). *Lecture Notes in Chemistry*, 67:139. 8
- [Schwarz et al., 2002] Schwarz, K., Blaha, P., and Madsen, G. K. H. (2002). *Comput. Phys. Commun.*, 147:71. 8
- [Schwarz et al., 1979] Schwarz, K., Neckel, A., and Nordgren, J. (1979). *J. Phys. F: Met. Phys.*, 9:2509. 11, 229
- [Schwarz and Wimmer, 1980] Schwarz, K. and Wimmer, E. (1980). *J. Phys. F: Met. Phys.*, 10:1001. 11, 229
- [Singh, 1989] Singh, D. (1989). *Phys. Rev. B*, 40:5428. 151
- [Singh and Nordström, 2006] Singh, D. J. and Nordström, L. (2006). *Planewaves, Pseudopotentials and the LAPW Method, 2nd ed.* Springer, Berlin. 8, 11, 158
- [Sjöstedt et al., 2000] Sjöstedt, E., Nordström, L., and Singh, D. J. (2000). *Solid State Commun.*, 114:15. 10, 151
- [Slater, 1951] Slater, J. C. (1951). *Phys. Rev.*, 81:385. 61, 141
- [Sofo and Fuhr, 2001] Sofo, J. and Fuhr, J. (2001). *see \$WIENROOT/SRC/aim\_sofo\_notes.ps.* 11, 174
- [Soler and Williams, 1989] Soler, J. M. and Williams, A. R. (1989). *Phys. Rev. B*, 40:1560. 11
- [Stahn et al., 2001] Stahn, J., Pietsch, U., Blaha, P., and Schwarz, K. (2001). *Phys. Rev. B*, 63:165205. 11, 142
- [Stephens et al., 1994] Stephens, P. J., Devlin, F. J., Chabalowski, C. F., and Frisch, M. J. (1994). *J. Phys. Chem.*, 98:11623. 138, 141
- [Sun et al., 2013] Sun, J., Haunschild, R., Xiao, B., Bulik, I. W., Scuseria, G. E., and Perdew, J. P. (2013). *J. Chem. Phys.*, 138:044113. 138, 139
- [Sun et al., 2015a] Sun, J., Perdew, J. P., and Ruzsinszky, A. (2015a). *Proc. Natl. Acad. Sci. U.S.A.*, 112:685. 138, 139
- [Sun et al., 2015b] Sun, J., Ruzsinszky, A., and Perdew, J. P. (2015b). *Phys. Rev. Lett.*, 115:036402. 8, 136, 138, 139
- [Tao and Mo, 2016] Tao, J. and Mo, Y. (2016). *Phys. Rev. Lett.*, 117:073001. 8, 138, 139
- [Tao et al., 2003] Tao, J., Perdew, J. P., Staroverov, V. N., and Scuseria, G. E. (2003). *Phys. Rev. Lett.*, 91:146401. 8, 138, 139

- [Terentjev et al., 2018] Terentjev, A. V., Constantin, L. A., and Pitarke, J. M. (2018). *Phys. Rev. B*, 98:214108. 68, 146
- [Tran, 2012] Tran, F. (2012). *Phys. Lett. A*, 376:879. 59
- [Tran et al., 2020] Tran, F., Bausesson, G., Carrete, J., Madsen, G., Blaha, P., Schwarz, K., and Singh, D. (2020). Shortcomings of meta-gga functionals. *Physical Review B*, 102:024407. 174
- [Tran and Blaha, 2009] Tran, F. and Blaha, P. (2009). *Phys. Rev. Lett.*, 102:226401. 8, 61, 62, 63, 86, 136, 138, 141
- [Tran and Blaha, 2011] Tran, F. and Blaha, P. (2011). *Phys. Rev. B*, 83:235118. 8, 56, 57, 85, 138, 141, 156, 157, 158
- [Tran et al., 2015a] Tran, F., Blaha, P., Betzinger, M., and Blügel, S. (2015a). *Phys. Rev. B*, 91:165121. 141
- [Tran et al., 2016] Tran, F., Blaha, P., Betzinger, M., and Blügel, S. (2016). *Phys. Rev. B*, 94:165149. 61
- [Tran et al., 2015b] Tran, F., Blaha, P., and Schwarz, K. (2015b). *J. Chem. Theory Comput.*, 11:4717. 61
- [Tran et al., 2006] Tran, F., Blaha, P., Schwarz, K., and Novák, P. (2006). *Phys. Rev. B*, 74:155108. 8, 11, 53, 147
- [Tran et al., 2018] Tran, F., Ehsan, S., and Blaha, P. (2018). *Phys. Rev. Materials*, 2:023802. 64
- [Tran et al., 2019] Tran, F., Kalantari, L., Traore, B., Rocquefelte, X., and Blaha, P. (2019). *Phys. Rev. Materials*, 3:063602. 68
- [Tran et al., 2007] Tran, F., Laskowski, R., Blaha, P., and Schwarz, K. (2007). *Phys. Rev. B*, 75:115131. 136
- [Tran et al., 2017] Tran, F., Stelzl, J., Koller, D., Ruh, T., and Blaha, P. (2017). *Phys. Rev. B*, 96:054103. 66, 67, 146
- [Trzhaskovskaya et al., 2001] Trzhaskovskaya, M., Nefedov, V., and Yarzhemsky, V. (2001). *Atom. Data Nucl. Data Tables*, 77:97--159. 212, 213
- [Trzhaskovskaya et al., 2006] Trzhaskovskaya, M., Nikulin, V., Nefedov, V., and Yarzhemsky, V. (2006). *Atom. Data Nucl. Data Tables*, 92:245--304. 212, 213
- [van Leeuwen and Baerends, 1994] van Leeuwen, R. and Baerends, E. J. (1994). *Phys. Rev. A*, 49:2421. 141
- [Van Voorhis and Scuseria, 1998] Van Voorhis, T. and Scuseria, G. E. (1998). *J. Chem. Phys.*, 109:400. 138, 139
- [Vinet et al., 1989] Vinet, P., Rose, J., Ferrante, J., and Smith, J. (1989). *J. Phys.: Condens. Matter*, 1:1941. 239
- [Vosko et al., 1980] Vosko, S. H., Wilk, L., and Nusair, M. (1980). *Can. J. Phys.*, 58:1200. 139, 142
- [Vydrov and Van Voorhis, 2010] Vydrov, O. A. and Van Voorhis, T. (2010). *J. Chem. Phys.*, 133:244103. 68, 146

- [Wei et al., 1985] Wei, S. H., Krakauer, H., and Weinert, M. (1985). *Phys. Rev. B*, 32:7792. 8
- [Weinert, 1981] Weinert, M. (1981). *J. Math. Phys.*, 22:2433. 8, 134
- [Weinert et al., 1982] Weinert, M., Wimmer, E., and Freeman, A. J. (1982). *Phys. Rev. B*, 26:4571. 8, 11, 134
- [Weintraub et al., 2009] Weintraub, E., Henderson, T. M., and Scuseria, G. E. (2009). *J. Chem. Theory Comput.*, 5:754. 56, 138, 141
- [Wellendorff et al., 2014] Wellendorff, J., Lundgaard, K. T., Jacobsen, K. W., and Bligaard, T. (2014). *J. Chem. Phys.*, 140:144107. 138
- [Wimmer et al., 1981] Wimmer, E., Krakauer, H., Weinert, M., and Freeman, A. J. (1981). *Phys. Rev. B*, 24:864. 8
- [Wu and Cohen, 2006] Wu, Z. and Cohen, R. E. (2006). *Phys. Rev. B*, 73:235116. 128, 136, 138, 141
- [Xue et al., 2018] Xue, K.-H., Yuan, J.-H., Fonseca, L. R., and Miao, X.-S. (2018). *Comput. Mater. Sci.*, 153:493. 65
- [Yanchitsky and Timoshevskii, 2001] Yanchitsky, B. Z. and Timoshevskii, A. N. (2001). *Comput. Phys. Commun.*, 139:235. 126
- [Yeh and Lindau, 1985] Yeh, J. J. and Lindau, I. (1985). *Atom. Data Nucl. Data Tables*, 32:1. 212
- [Yu et al., 1991] Yu, R., Singh, D., and Krakauer, H. (1991). *Phys. Rev. B*, 43:6411. 11, 134
- [Zhang and Yang, 1998] Zhang, Y. and Yang, W. (1998). *Phys. Rev. Lett.*, 80:890. 138, 141
- [Zhao and Truhlar, 2008] Zhao, Y. and Truhlar, D. G. (2008). *J. Chem. Phys.*, 128:184109. 138, 141