

Lithium ELNES with WIEN2k

September 11, 2018

Contents

1	Tools	1
2	Setup	2
3	Convergence	5
3.1	Cell parameters:	6
3.2	K point and RKMax convergence	7
4	TELNES3	8
4.1	Monopole effects	9
5	Core Hole	10
6	Density Calculations	13
7	Common Errors/Issues Encountered in Wien2k	15
7.1	Setting RMT/RKMax	15
7.2	Ghostbands	16
7.3	NN in Optimization	17
7.4	GMax Value less than Gmin	18
7.5	Ordering	18

1 Tools

Here's a list of software I use to run simulations. I've put some installation instructions for linux as needed.

- Wien2k - If you are reading this guide, you either already have it installed somewhere, or should figure out how to do that before buying a license.
- VESTA: <http://jp-minerals.org/vesta/en/download.html>. Download the .rpm file, install it with your package manager, eg. "sudo apt-get vesta...rpm"

- Critic2: <https://github.com/aoterodelaroza/critic2>. Download the zip from GitHub, unzip where you want to install it eg `~/Programs` or something), install (using `dnf/apt-get`) `autoconf`, `automake`, and your favourite flavour of `fortran`. Then run the 4 commands from the readme: “`autoreconf -i`”, “`./configure`”, “`make`”, and “(sudo) `make install`”. You may or may not need the `sudo` for the last one.

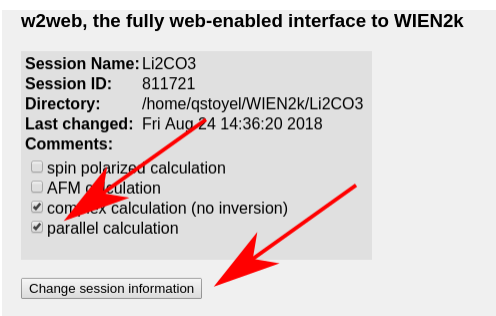
2 Setup

We need a couple things to get started for ELNES simulations. Foremost is a crystal structure. You can either get this from the literature, XRD, or alternatively Materials Project: <https://materialsproject.org/>. Download a cif (the primitive cell typically) or enter the coordinates directly into the wien2k struct gen tool.

Make a new Wien2k session:



Create/change a working directory, and change the session information for parallel calculation.



You will also need to make a “.machines” file which you can either steal from one of my directories, look in the user guide and make your own, cut and paste the one from below, or wait for w2web to automatically generate one at some point after it inevitably crashes on something. Sample .machines file:

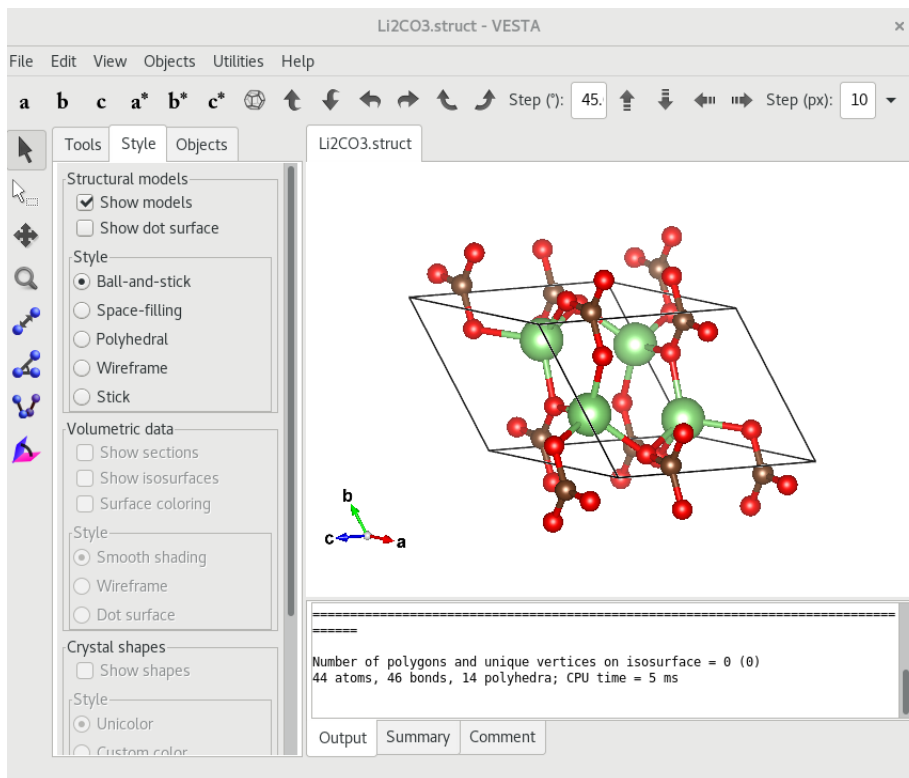
```
#####
#This is a valid .machines file
#
```

```

granularity:1
1:localhost #as many of these lines as you want cpu cores running
1:localhost #ideally pick a multiple of kpoint number
1:localhost #so don't use 5 cores for 13 k points...
1:localhost
1:localhost
1:localhost

```

Next, go make a struct file, with struct gen, either by importing the cif, or entering the positions manually. Use VESTA (drag n' drop the .struct file) to make sure that the structure is what you'd expect. In our Li_2CO_3 case this looks like:



The next step is initialization. This is best done via the command line as it give a little more flexibility and intuition than in w2web, and will need to be run multiple times. Run **init_lapw** and go through the following steps:

- **setrmt.** Sets the muffin tin size on the atoms. Reduce the sphere size by 0% using either old or new scheme and accept, it doesn't matter much as these will be reset properly later.
- **nn.** Checks for overlapping muffin tins. Enter "2.0", close the first file and use the new NN file if suggested, run **nn** with 2.0 again, look at how much "wiggle room" you have on the spheres, by comparing the "sums to" and the "nn-dist" for every atom, see below:

```

Terminal
File Edit View Search Terminal Help
qstoyel@Li2C03$ init_lapw
next is setrat
Automatic determination of RMTs. Please specify the desired RMT reduction
compared to almost touching spheres.
Typically, for a single calculation just hit enter, for force minimization
use 1-5; for volume effects you may need even larger reductions.

Enter reduction in %
0
Use old or new scheme (o/N)
n
specify nn-bondlength factor: (usually=2) [and optionally dlimit, dstmax (about
1.d-5, 20)]
DSTMAT: 20.000000000000000
iix,iiz 3 3 27.818223000000000 27.
818223000000000 35.986764000000001
ATOM 1 Li ATOM 8 0
RMT( 1)=1.63000 AND RMT( 8)=1.27000
SUMS TO 2.90000 LT. NN-DIST= 3.63663
ATOM 2 Li ATOM 8 0
RMT( 2)=1.63000 AND RMT( 8)=1.27000
SUMS TO 2.90000 LT. NN-DIST= 3.63663

```

- **sgroup.** Verifies the space group. Again, accept any changes the program makes, these steps are all about reducing the cell to the smallest unit cell. Again, the files can be largely ignored at this point, with the exception of indication of a Bravais lattice change, in which case, accept the new struct file. If so, nn and sgroup will run again with “nice” results.
- **symmery.** Generates all the symmetry operations. Run it and continue (enter “c”)
- **lstart.** Set spin state, select which XC kernel, define cutoff between core and valence states. Accept default spins (up, the no spin case), unless there is a transition metal involved. Select GGA PBE as the XC potential, again, unless there is reason to suspect otherwise. Picking the energy is the most important part of this first run init_lapw: the aim is to get the Li 1s states to be treated as core states. They typically have energies of $\sim -3.7\text{Ry}$, so try with -3.5 Ry to make sure they will be treated as core states and look in case.outputst (the file that pops up), for the following lines for the lithium atom and look at the 1S states:

	E-up (Ry)	E-dn (Ry)	Occupancy	q/sphere	core-state
1S	-3.801947	-3.785288	1.00	1.00	0.9859 T
1S	-3.801947	-3.785288	1.00	1.00	0.9859 T
2S	-0.236699	-0.003313	1.00	0.00	0.0468 F
2S	-0.236699	-0.003313	1.00	0.00	0.0468 F

These indicate the core states (T/F), their energy levels (in this case $\sim -3.8\text{eV}$) and how much the electrons in these states live in the muffin tins (0.9859). As this is less than 1, it means a lot of 1S lithium electron is leaking out of the muffin tins, which is why there should now be all kinds of warnings popping up. So go ahead and “ctrl-c” out of init_lapw.

To fix the leakage problem, the Lithium muffin tins need to be bigger. Ideally, they should be just big enough to hold all of the 1S electrons, without making them too different from the other muffin tins, as the larger this difference, the harder things get to calculate/converge. In this case, try $\text{Li}=1.8$, $\text{C}=1.2$, $\text{O}=1.22$ and try init_lapw again, if that still didn’t work (lstart still had leakage errors), keep going until it does, in this case $\text{RMT}_{\text{Li}} \approx 2\text{RMT}_{\text{C}}$ and will cause errors

(ghostbands) further along. The solution here is to try and minimize the leakage (and ignore the warnings) and acknowledge that it might be unavoidably causing artifacts. For this case I chose $\text{Li}=1.8$, $\text{C} = 1.17$, $\text{O}=1.26$. Alternatively, pick a nicer structure to analyze. Once the spheres are set in the struct file, rerun **init_lapw**:

- **setrmt**: Setrmt will try to reset the muffin tins to the defaults, make sure to discard these (enter d)
- **nn** Make sure you don't get errors, and that everything is as tight as it can be, in this case the Oxygen-carbon spacing is the limiting factor.
- **sgroup** Should run fine.
- **symmery** Should run fine.
- **lstart** Now that the lithium 1S states are well contained, core states can be selected based on containment instead of energy, meaning, the higher energy states (2S, P) of other elements can still be treated as valence. Entering 0.99 should be sufficient here, but make sure to verify that the lithium states are still core states.
- **kgen** Set the RkMax value in case.in1.st:

```

GNU nano 2.5.3 File: Li2CO3.in1.st
%FFIL EF= 0.50000 (WFFIL, WFPRI, ENFIL, SUPWF)
7.00 10 4 (R-MT*K-MAX; MAX L IN WF, V-NMT)
0.30 1 0 (GLOBAL E-PARAMETER WITH n OTHER CHOICES, global APW/LAPW)
0 0.30 3 0 0.000 CONT 1
0.30 3 0 (GLOBAL E-PARAMETER WITH n OTHER CHOICES, global APW/LAPW)
0 -0.70 0.002 CONT 1
0 0.30 0.000 CONT 1
1 0.30 0.000 CONT 1
0.30 3 0 (GLOBAL E-PARAMETER WITH n OTHER CHOICES, global APW/LAPW)
0 -1.55 0.002 CONT 1
0 0.30 0.000 CONT 1
1 0.30 0.000 CONT 1
0.30 3 0 (GLOBAL E-PARAMETER WITH n OTHER CHOICES, global APW/LAPW)
0 -1.55 0.002 CONT 1
0 0.30 0.000 CONT 1
1 0.30 0.000 CONT 1
K-VECTORS FROM UNIT:4 -11.0 1.5 101 emin / de (emax=Ef+de) / nband

```

and then pick a k_point number. Both of these values should initially be taken for fast convergence, in this case I chose $\text{RkMax}=6.0$, and 8 k points (Li₂CO₃ is an insulator, for metals 1000 k points is a good starting point).

- **Dstart**: make sure to pick non spin polarized, unless there is reason to believe otherwise (is there a transition metal in the sample?)

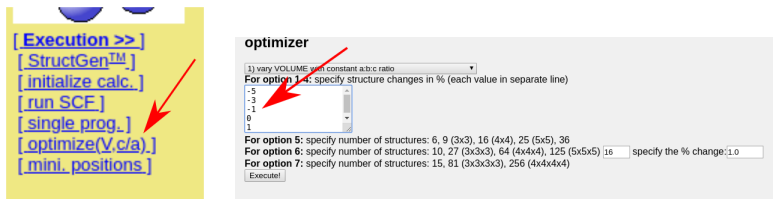
Assuming there were no warnings in the final run through **init_lapw**, the case can begin to be converged.

3 Convergence

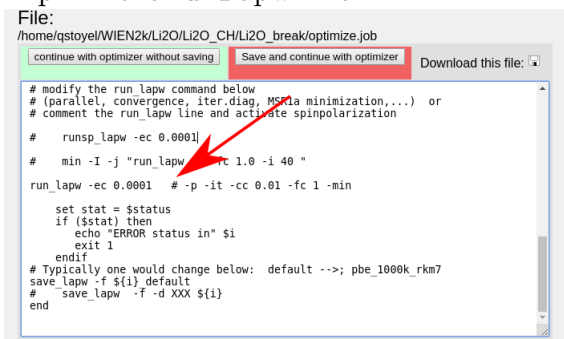
Ideally, every variable should be converged regarding the simulation. Typically this is cell parameters, k points and Rkmax. The first step is to just make sure the calculation converges, running it with a small RKmax and few kpoints and making sure it finishes without error.

3.1 Cell parameters:

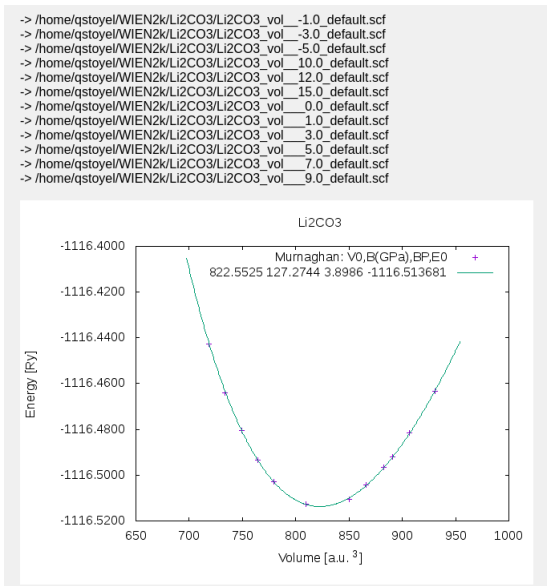
This process works best from W2Web, as described in the tutorials and using the muffin tins from **setrmt** reduced by a healthy percentage ($\sim 5\text{-}10\%$) to avoid nn errors (you may need to rerun **init_lapw**). As a number (5-11+) of calculations are run in this process, a low number of k points and RKMax values is ideal here. For Li_2CO_3 , I used 16 kpoints and an RKmax of 6. In the x “optimize” tab, choose what you want to optimize, the first option (volume) works well, unless there are suspicions otherwise. Enter a range of values of test volumes, see picture:



Also make sure to edit “optimize.job” to enable parallization by moving the “#” to after the “-p” in the run_lapw line:



You can then “run optimize.job” from w2web. This job is okay to run in the background, which means the browser can be closed without the job stopping. When it is done, plot the “Energy vs Volume,” which should look something like this:



In the case of Li_2CO_3 and the cif from materials project, quite a range of volume options were needed to locate the minimum. The graph indicates that a 3-4% increase should correspond to the optimized structure. To use this structure, search the case directory for all the struct files and rename the appropriate one to case.struct, for Li_2CO_3 this was “ $\text{Li}_2\text{CO}_3\text{.vol_3.0.struct} \rightarrow \text{Li}_2\text{CO}_3\text{.struct}$ ”. Alternatively, cut and paste the lattice parameters from this file into the w2web structgen tool. Finally, rerun init_lapw and readjust sphere sizes as necessary to account for the increase (or decrease) in cell size.

3.2 K point and RKMax convergence

To converge these parameters, again start with very low values and then increase them, checking the total energy to determine when they are converged. Generally k points are easier to converge, so start with them and then move on to RKMax. The procedure for converging both of these values is:

- set/increase kpoints or RKMax, either by re-running “x kgen” or editing case.in1 and case.in1.st.
- Run the scf cycle using “run_lapw -p -NI”, the NI flag means it will continue from where the previous calculation left off which saves time. I would still run your final choice from scratch though.
- Check the energy in case.scf. To do this, find it in “scf files” on w2web and use ctrl+f in your browser to search the file for “:ene” , which should appear in a line that looks like:

```
:ENE : ***** TOTAL ENERGY IN Ry = -1116.50733157
```

There will be one of these lines for each scf cycle, so find the last one in the document and note the energy.

- loop through the first 3 steps until the energy no longer changes significantly when you increase the kpoints/RKMax. A table is useful here to track these effects eg:

kpoints	RKMax	Energy
8	7.0	-1116.5073
16	7.0	-1116.4971
32	7.0	-1116.4975
32	8.0	-1116.5038
32	9.0	-1116.5045

4 TELNES3

Once the calculation is converged, ELNES can be calculated. There is a good deal of useful information and a complete description of the input file in the wien2k userguide, which should solve most issues. Again, there is a large list of parameters that must be set for in order to obtain reasonable results. It is also worth converging Kpoints and RKMax against the spectra as well.

The majority of the important parameters need to be set in the **case.innes** file, which is easiest through w2web. Choose the right atom (in this case Li1) for the edge, and the right atomic numbers:

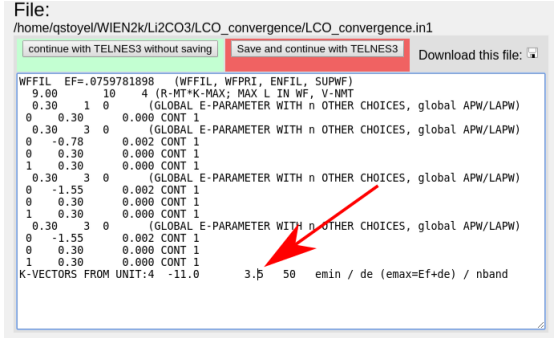
Edge	n	l
K	1	0
L1	2	0
L23	2	1
M45	3	2

Next, set the edge onset, edge values can be found at http://www.kayelaby.npl.co.uk/atomic_and_nuclear_physics/4_2/4_2_1.html as well as at a number of other locations. Set the beam energy to it's correct value, same goes for the collection and convergence angles, although TELNES is relatively robust to these: eg 5mrad produces very similar results to 1 mrad.

Set the energy grid to a large range of values, eg -20-50eV so you can see all of the features that might appear. The defaults for the remaining values should be fine.

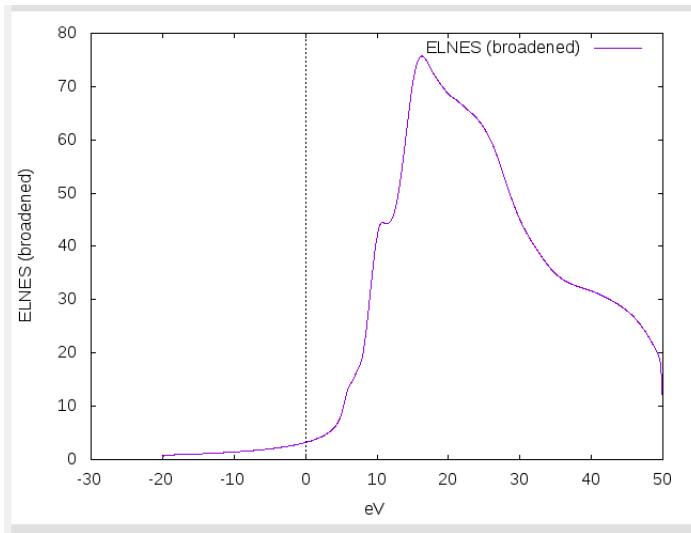
In addition to the case.innes parameters, increase the number of kpoints, to at least double, or $10\times$, so that there is less doubt about this being converged, use **x kgen** for this.

Increase the upper energy limit in case.in1 from 1.5 to $\sim 2-3.5$, see picture. This value defines how many higher energy states are included in Ry ($1 \text{ Ry} \approx 13.6 \text{ eV}$, $1.5 \text{ Ry} \approx 20 \text{ eV}$). Therefore, to obtain the correct ELNES for features more than 20eV from the onset, this should be increased to match.



`x lapw1 -p`, `x qtl -telnes -p`, and `x telnes3` can then all be run in succession. These are relatively slow commands, so to avoid waiting on them, run all three in sequence from the command line: `$ x lapw1 -p && x qtl -telnes -p && x telnes3`.

Once `telnes3` is finished, edit `case.inb` and play with the spectrometer broadening on the last line, before clicking “x broadening” to generate the final spectra. To experiment with different broadening values, just repeat these steps. Sometimes `w2web` gets stuck and stops re-broadening the spectra, in which case, just delete the `case.broadspec` file and run “x broadening” again. Plot the spectrum in a new tab (ctrl+click on “plot”) for easy comparison to new spectra without having to save it. Rerun all the TELNES steps, but with a higher/lower number of kpoints to confirm that that is not effecting the spectra. At this point you can also uncheck the “calculate and write DOS” checkboxes in `case.innes`, which allow you to rerun `telnes` without needing to rerun `lapw1` each time (unless you change the k points/`case.in1` file). If everything went well, the ELNES should look something like:



4.1 Monopole effects

Because the lithium 1S state is quite delocalized, it is susceptible to monopole effects. These cause artifacts in the spectra resulting from non orthogonal states at the muffin tin boundary. Fortunately, it is easy to check for them. Return to the `case.innes` file and change the interaction order to 0.

Title: Lithium K edge in Li2CO3 no hole

Atom: 1: Li1 **Edge:** use n and l (n=1 l=0)

Edge onset: 55.00 eV **Beam energy:** 300 keV

Energy grid: -20.0000 eV to 50.0000 eV in steps of 0.0500 eV

Collection s.a.: 5.00 mrad **Convergence s.a.:** 1.87 mrad

Spectrometer broadening: 0.50 eV **Q-mesh:** NR=5 NT=2

Advanced settings:

Branching ratio: (statistical if empty)

Spinorbit splitting of core state (eV): (calculated if empty)

☐ **Orientation sensitive:** α=°, β=°, γ=°

Integrate over equivalent atoms: to (all eq. atoms if empty)

Detector position: θ_x 0.000 mrad, θ_y 0.000 mrad

Modus: energy

Initialization: ☒ Calculate DOS ☒ write DOS

☒ Calculate rotation matrices ☒ write rotation matrices

Verbosity: basic **File headers:** Write headers (default)

Interaction potential: relativistic (recommended)

Q-grid: U uniform θ_0= (not used for uniform grid)

Interaction order: λ=0 **Final state selection rule:** L=| +/- 1 (default)

☐ **Extend potential beyond Rmt:** rmax= bohr

☐ **Set Fermi energy manually:** EF= Ry

☐ **Read core state wavefunction:** filename= case.cwf

☐ **Read final state wavefunctions:** filename= case.finalwf

☐ **Calculate DOS only**

☐ **NBTOT:** 200

save

Telnes now only calculates the monopole component, look in case.outputtelnes to confirm this. Compare the monopole contribution to the full spectra, it should be much smaller ($\sim 100\times$). If it is not, set the interaction order to 1 to enforce dipole selection and use that for all future spectra.

5 Core Hole

To introduce a core hole, start an entirely new case, and copy only the struct file. Because of the periodic boundary conditions, it is sometimes necessary to use a supercell to avoid core holes interacting with themselves in neighbouring cells. This depends largely on the size of the original unit cell, for a single atom case (metallic lithium) a $3\times 3\times 3$ cell is required, for large calcium phosphates, no cell is needed.

Li2CO3 is a mid sized cell. To verify that there is no interaction, the single cell case with a core hole will be compared to a $2\times 1\times 1$ supercell. If there is any significant variation in results a $2\times 2\times 2$ supercell will be necessary.

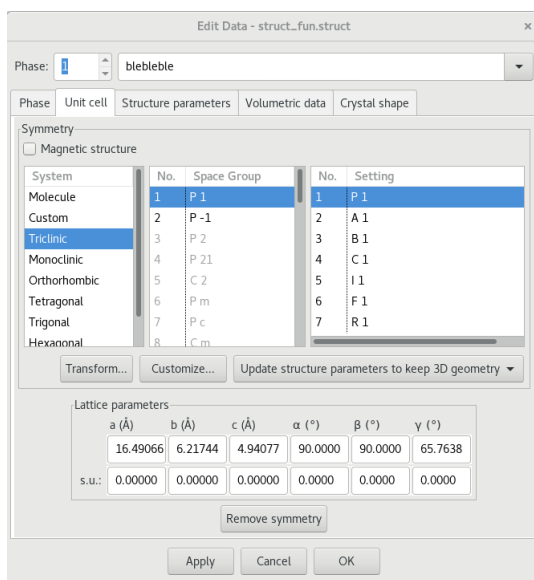
With all core holes, it is essential to remove the symmetry in the unit cell, so there is only a single hole per cell:

VS

Inequivalent Atoms: 12
Atom 1: Li Z=3.0 RMT=2.0000
 Pos 1: x=0.19885150 y=0.16104400 z=0.05090650
Atom 2: Li Z=3.0 RMT=2.0000
 Pos 1: x=0.80114850 y=0.33895600 z=0.05090650

Inequivalent Atoms: 4
Atom 1: Li 1 Z=3.0 RMT=2.0000
 Pos 1: x=0.19885150 y=0.16104400 z=0.05090650
 Pos 2: x=0.80114850 y=0.33895600 z=0.05090650
 Pos 3: x=0.80114850 y=0.83895600 z=0.94909350
 Pos 4: x=0.19885150 y=0.66104400 z=0.94909350

To achieve this, the cell must be reduced to a P1 space group. This can be done in VESTA. Import the struct file and set the space group to P1, under “Edit → Edit Data → Unit Cell.” Select space group No. 1, and apply.



To save these changes select “File → Export data” and save the structure as a .cif. Because there are some bugs in the Wien2k-VESTA interface, the cif needs to be edited a bit manually as well. Open the cif and delete the following line in the initial block of data:

```
_space_group_name_H-M_alt          'P 1'
```

and replace it with this line at the top of that block:

```
_symmetry_space_group_name_H-M     'P 1'
```

Running **Cif2struct name_of_cif.cif** should then be able to generate a valid struct file, which should be renamed as case.struct.

In the new struct file, add a “1” to the end of an atom (AND DELETE A SPACE). The file should look like this:

Delete space from here!

```

blebleble
P
48
RELA
31.162150 11.749259 9.336702 90.000000 90.000000 65.763760
ATOM 1: X=0.09942600 Y=0.16104400 Z=0.05090700
MULT= 1 ISPLIT= 8
Li1 NPT= 781 R0=0.00010000 RMT= 1.5900 Z: 3.0
LOCAL POT MATRIX: 1.0000000 0.0000000 0.0000000
0.0000000 1.0000000 0.0000000
0.0000000 0.0000000 1.0000000
ATOM -2: X=0.40057400 Y=0.33895600 Z=0.05090700
MULT= 1 ISPLIT= 8
  
```

Add "1" here.

Now run **init_lapw** and accept all of the defaults proposed by **nn**, which may require cycling through **nn** a couple times. **sgroup** should not have any suggestions here, if it does, something has gone wrong and it is trying to restore all of the symmetry. At the end of **init_lapw**, make sure to double check that the struct file is still what it should be.

When the struct file is ready, initialize use the converged parameters (RMT, RKMax) from the single cell case. After **init_lapw**, insert a core hole into the cell by editing **case.inc** and **case.inm**:

```

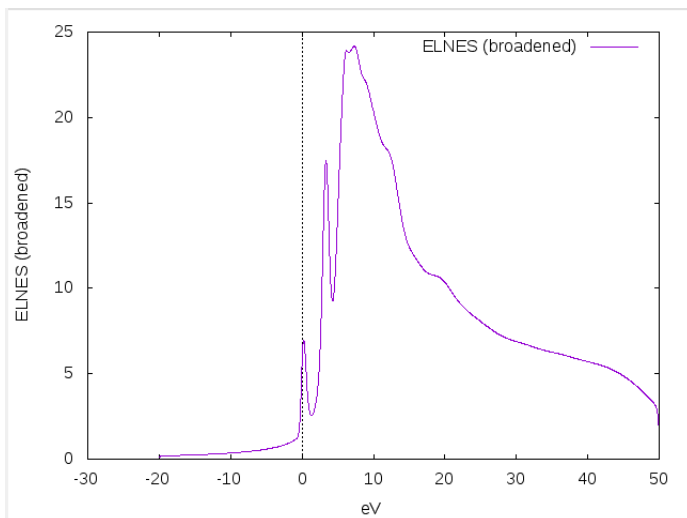
GNU nano 2.5.3      File: Li2C03 hole sc.inc      Modified
1 0.00 0 NUMBER OF ORBITALS (EXCLUDING SPIN), SHIFT, IPRINT
1,-1,1 ( N,KAPPA,OCCUP)
1 0.00 0 NUMBER OF ORBITALS (EXCLUDING SPIN), SHIFT, IPRINT
1,-1,2 ( N,KAPPA,OCCUP)
1 0.00 0 NUMBER OF ORBITALS (EXCLUDING SPIN), SHIFT, IPRINT
1,-1,2 ( N,KAPPA,OCCUP)
1 0.00 0 NUMBER OF ORBITALS (EXCLUDING SPIN), SHIFT, IPRINT
1,-1,2 ( N,KAPPA,OCCUP)
1 0.00 0 NUMBER OF ORBITALS (EXCLUDING SPIN), SHIFT, IPRINT
1,-1,2 ( N,KAPPA,OCCUP)
1 0.00 0 NUMBER OF ORBITALS (EXCLUDING SPIN), SHIFT, IPRINT
1,-1,2 ( N,KAPPA,OCCUP)
1 0.00 0 NUMBER OF ORBITALS (EXCLUDING SPIN), SHIFT, IPRINT
1,-1,2 ( N,KAPPA,OCCUP)
1 0.00 0 NUMBER OF ORBITALS (EXCLUDING SPIN), SHIFT, IPRINT
1,-1,2 ( N,KAPPA,OCCUP)
  
```

In **case.inc**, a hole is inserted by changing the occupancy of an orbital. Each set of lines corresponds to the core states of each atom defined in the struct file. This is why it is important to be able to treat the lithium states as core states, otherwise it would not be possible to insert a hole. This is also why it is important to break the symmetry in the unit cell as if not, this would insert holes into multiple atoms. In the **case.inm** file, the excited electrons need to be added to the background charge:

```

GNU nano 2.5.3      File: Li2C03 hole sc.inm
MSR1 -1.00 YES (BROYD/PRATT, BG charge (-1 for core hole), norm)
0.20 mixing FACTOR for BROYD/PRATT scheme
1.00 1.00 PW and CLM-scaling factors
9999 8 idum, HISTORY
  
```

The value is negative as this operation adds charge, not electron number. Once the background charge is added, execute **run_lapw -p** and calculate the ELNES. If done correctly, your **elnes** should look something like this:



To make a supercell, run **x supercell** in the directory and make a $2 \times 1 \times 1$. Then use the same procedure to keep the symmetry broken: import to vesta, select P1 spacegroup, save cif, edit spacegroup line in cif, cif2struct, relabel an atom with a number, and then run **init_lapw**. Reduce K points by a factor of 2 (the unit cell is now twice as big), and keep the RKMax from earlier. Insert a hole in case.inc, case.inm and then **run_lapw**.

6 Density Calculations

Density calculations can be done using Critic2. Documentation for the code, while very complete, is just a giant txt file. The code takes three inputs: the case.struct file, the case.clmsum file, and an input parameters file “more.cri”. A sample .cri file is below:

```
crystal ./Li2CO3.struct
load ./Li2CO3.clmsum ./Li2CO3.struct
auto
integrals gauleg 50 50 cp 1 verbose
sphereintegrals gauleg 50 50 cp 1 R0 1d-1 REND 2.0
```

The first line inputs the crystal structure, and then tells the code where to find the clmsum/struct files in the second line.

The third line determines the location of all the critical points.

The 4th line calculates the electron population inside the atomic basin surrounding critical point 1. “Gauleg” is a integration option, look in the critic documentation for more options. The two 50’s are sampling parameters, increase/decrease these for more/less accurate calculations.

The 5th line calculates the electron population in spheres surrounding critical point 1. “gauleg 50 50” mean the exact same as on the previous line, R0 is the starting radius, 1d-1

is the radius step size, REND is the final radius (in bohr).

The program is run using `$ critic2 ; more.cri ; case.cro` in the command line in the case directory. The final file (case.cro) is just where the output is stored and can be called anything. To determine which critical points are needed for the integrals, look at the complete cp list in the output file:

```
* Complete CP list
# (x symbols are the non-equivalent representative atoms)
# cp ncp typ position (cryst. coords.) op. (lvec+cvec)
x 1 1 n 0.19885150 0.16104400 0.05090650 1 0.0 0.0 0.0
2 1 n 0.69885150 0.16104400 0.55090650 1 0.5 0.0 0.5
3 1 n 0.30114850 0.33895600 0.05090650 2 1.0 0.0 0.0
4 1 n 0.30114850 0.33895600 0.55090650 2 0.5 0.0 0.5
5 1 n 0.80114850 0.83895600 0.94909350 3 1.0 1.0 1.0
6 1 n 0.30114850 0.83895600 0.44909350 3 0.5 1.0 0.5
7 1 n 0.19885150 0.66104400 0.94909350 4 0.0 0.0 1.0
8 1 n 0.69885150 0.66104400 0.44909350 4 0.5 0.0 0.5
9 2 n 0.00000000 0.75000000 0.43496300 1 0.0 0.0 0.0
10 2 n 0.50000000 0.75000000 0.93496300 1 0.5 0.0 0.5
11 2 n 0.00000000 0.25000000 0.56503700 3 0.0 1.0 1.0
12 2 n 0.50000000 0.25000000 0.06503700 3 0.5 1.0 0.5
x 13 3 n 0.00000000 0.75000000 0.18014600 1 0.0 0.0 0.0
14 3 n 0.50000000 0.75000000 0.68014600 1 0.5 0.0 0.5
15 3 n 0.00000000 0.25000000 0.81985400 3 0.0 1.0 1.0
16 3 n 0.50000000 0.25000000 0.31985400 3 0.5 1.0 0.5
x 17 4 n 0.64687450 0.68525300 0.06479250 1 0.0 0.0 0.0
18 4 n 0.14687450 0.68525300 0.56479250 1 -0.5 0.0 0.5
19 4 n 0.35312550 0.81474700 0.06479250 2 1.0 1.0 0.0
20 4 n 0.85312550 0.81474700 0.56479250 2 1.5 1.0 0.5
21 4 n 0.35312550 0.31474700 0.93520750 3 1.0 1.0 1.0
22 4 n 0.85312550 0.31474700 0.43520750 3 1.5 1.0 0.5
23 4 n 0.64687450 0.18525300 0.93520750 4 0.0 -1.0 1.0
24 4 n 0.14687450 0.18525300 0.43520750 4 -0.5 -1.0 0.5
x 25 5 b 0.50000000 0.75000000 0.84824527 1 0.0 0.0 0.0
26 5 b 0.00000000 0.75000000 0.34824527 1 -0.5 0.0 -0.5
27 5 b 0.50000000 0.25000000 0.15175473 3 1.0 1.0 1.0
28 5 b 0.00000000 0.25000000 0.65175473 3 0.5 1.0 1.5
x 29 6 b 0.05015081 0.72791128 0.47968374 1 0.0 0.0 0.0
30 6 b 0.55015081 0.72791128 0.97968374 1 0.5 0.0 0.5
31 6 b 0.94984919 0.77208872 0.47968374 2 1.0 1.0 0.0
32 6 b 0.44984919 0.77208872 0.97968374 2 0.5 1.0 0.5
33 6 b 0.94984919 0.27208872 0.52031626 3 1.0 1.0 1.0
34 6 b 0.44984919 0.27208872 0.02031626 3 0.5 1.0 0.5
35 6 b 0.05015081 0.22791128 0.52031626 4 0.0 -1.0 1.0
36 6 b 0.55015081 0.22791128 0.02031626 4 0.5 -1.0 0.5
```

There are a number of similar lists which can be used to identify which atom is of interest. The critical point should be a nuclear critical point (typ = n), and as a double check, the coordinates should match up to those in the struct file. The name should also match, in this case, the ncp 1 corresponds to "Li1."

The electron populations are located towards the bottom of the document. The first case looks like this:

```
* Integrated atomic properties
# (See key above for interpretation of column headings.)
# Integrable properties 1 to 3
# Id cp ncp Name Z mult Volume Pop Lap
1 1 1 Li1 3 8 2.07816976E+01 2.10985798E+00 -1.48481596E-02
-----
Sum 66253581E+02 1.88638E+01 -1.18785277E-01
End INTEGRALS--2018/9/10, 13:55:35.812
```

The two values indicated by the arrows are the volume of the basin and how many electrons are inside. The second population is calculated directly below this in a table:

#	r (bohr)	Eval/ray	Int. r_err	Volume	Charge	Lap
1.000000E-01	51	0.000000E+00	4.18879020E-03	3.75537284E-02	-1.76191919E+04	
1.030722E-01	51	0.000000E+00	4.58684179E-03	4.05781583E-02	-1.92932037E+04	
1.062389E-01	51	0.000000E+00	5.02271935E-03	4.38291944E-02	-2.11262769E+04	
1.095028E-01	51	0.000000E+00	5.50001740E-03	4.73218792E-02	-2.31335258E+04	
1.128670E-01	51	0.000000E+00	6.02267204E-03	5.10720352E-02	-2.53315014E+04	
1.163345E-01	51	0.000000E+00	6.59499341E-03	5.50962813E-02	-2.77383271E+04	
1.199086E-01	51	0.000000E+00	7.22170124E-03	5.94120446E-02	-3.03738486E+04	
1.235925E-01	51	0.000000E+00	7.90796373E-03	6.40375705E-02	-3.32597977E+04	
1.273895E-01	51	0.000000E+00	8.65944026E-03	6.89919276E-02	-3.64199710E+04	
1.313032E-01	51	0.000000E+00	9.48232796E-03	7.42950086E-02	-3.98804265E+04	
1.353372E-01	51	0.000000E+00	1.03834129E-02	7.99675269E-02	-4.36696987E+04	
1.394951E-01	51	0.000000E+00	1.13701259E-02	8.60310059E-02	-4.78190333E+04	
1.437807E-01	51	0.000000E+00	1.24506042E-02	9.25077640E-02	-5.23626453E+04	
1.481980E-01	51	0.000000E+00	1.36337579E-02	9.94208912E-02	-5.73380013E+04	
1.527510E-01	51	0.000000E+00	1.49293441E-02	1.06794219E-01	-6.27861279E+04	
1.574439E-01	51	0.000000E+00	1.63480471E-02	1.14652280E-01	-6.87519506E+04	
1.622809E-01	51	0.000000E+00	1.79015663E-02	1.23020262E-01	-7.52846641E+04	
1.672666E-01	51	0.000000E+00	1.96027131E-02	1.31923951E-01	-8.24381380E+04	
1.724054E-01	51	0.000000E+00	2.14655161E-02	1.41389658E-01	-9.02713607E+04	
1.777021E-01	51	0.000000E+00	2.35053372E-02	1.51444148E-01	-1.12192227E+05	
1.831616E-01	51	0.000000E+00	2.57389980E-02	1.62114541E-01	-1.15295119E+05	
1.887887E-01	51	0.000000E+00	2.81849188E-02	1.73428212E-01	-1.18366143E+05	
1.945888E-01	51	0.000000E+00	3.08632701E-02	1.85412675E-01	-1.21397546E+05	
2.005670E-01	51	0.000000E+00	3.37961393E-02	1.98095452E-01	-1.24378524E+05	
2.067289E-01	51	0.000000E+00	3.70077126E-02	2.11503930E-01	-1.27298099E+05	
2.130801E-01	51	0.000000E+00	4.05244747E-02	2.25665200E-01	-1.30144928E+05	
2.196265E-01	51	0.000000E+00	4.43754270E-02	2.40665887E-01	-1.32907319E+05	
2.263739E-01	51	0.000000E+00	4.85923269E-02	2.56351963E-01	-1.35573291E+05	
2.333287E-01	51	0.000000E+00	5.32099497E-02	2.72928545E-01	-1.38120597E+05	
2.404971E-01	51	0.000000E+00	5.82663750E-02	2.90359680E-01	-1.40566850E+05	
2.478857E-01	51	0.000000E+00	6.38033013E-02	3.08668113E-01	-1.42869529E+05	
2.555014E-01	51	0.000000E+00	6.98663897E-02	3.27875051E-01	-1.45026087E+05	
2.633510E-01	51	0.000000E+00	7.65056402E-02	3.47999905E-01	-1.47024025E+05	
2.714410E-01	51	0.000000E+00	8.37750041E-02	3.69060031E-01	-1.48850977E+05	
2.797811E-01	51	0.000000E+00	9.17368358E-02	3.91070450E-01	-1.50494805E+05	
2.883767E-01	51	0.000000E+00	1.00454387E-01	4.14043576E-01	-1.51943699E+05	
2.972363E-01	51	0.000000E+00	1.10000348E-01	4.37988926E-01	-1.53186270E+05	
3.063681E-01	51	0.000000E+00	1.20453441E-01	4.62912837E-01	-1.54211657E+05	

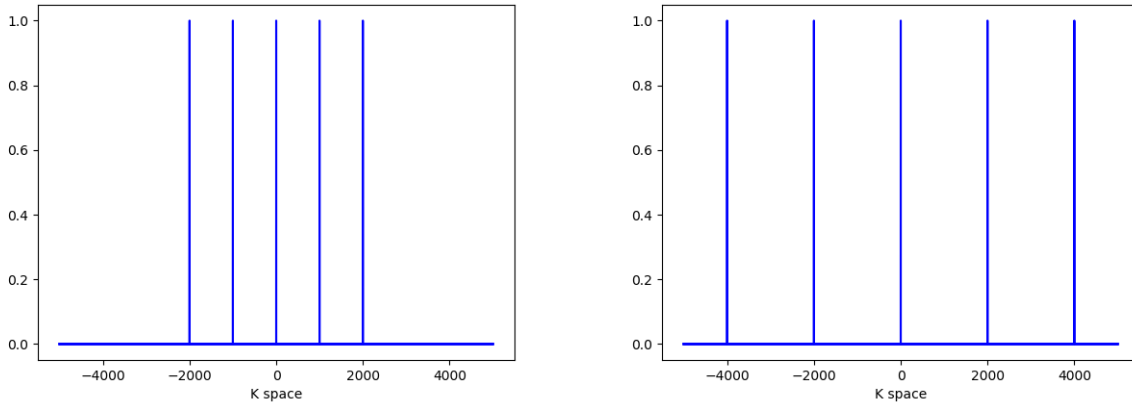
By matching to the volume value, relatively comparable values should be attainable, in the no hole case. The reason for performing two types of population calculation is due to the fact that the atomic basins can become ill defined when a core hole is inserted. This is revealed when the volume of the basin is dramatically larger ($> 20\%$) than the no hole case. In these situations, a comparison of the sphere integrals at the same volume (set by the basin volume in the no hole case) is more appropriate.

Once density calculations are performed for both hole and no-hole cases, core hole screening can be calculated by comparing the values. For example if the no-hole population is: 2.17, and the core hole population is 1.5, the hole would be screened by: $(1.5 + 1) - 2.17 = 0.33$ electrons. The +1 is to account for the excited electron. To implement this effect, a third, final supercell calculation is required, with a reduced hole inserted. case.inc would need an occupancy of 1.33 and case.inm would need a background charge of -0.67.

7 Common Errors/Issues Encountered in Wien2k

7.1 Setting RMT/RKMax

The RKMax value is a little obscure and takes some getting used to. It is defined as the Muffin Tin Radii \times Maximum K point vector. The maximum k point corresponds to the highest frequency plane wave used in the calculation. RKMax defines a cutoff for sampling k space, based on the size of each atom. A visualization of this effect is depicted below:



Two case, both with 5 kpoints and different RKMax's. The plot on the left depicts a smaller RKMax relative to the one on the right.

Increasing RKMax allows for higher frequency (more precise) plane waves to be used in the basis set. These high frequency terms are however more computationally costly (CPU requirements scale as RKMax^3) and less essential for describing large features. This is why the RKMax includes the Muffin Tin radius: large atoms only require lower frequency plane waves. Each atom has an effective RKMax which is relative to the ratio of its radius and that of the smallest atom. Choosing very different muffin tin radii for different elements leads to issues as it becomes more difficult to align the surface features of each atom.

Increasing RKMax arbitrarily does not solve the problem either, as: 1. “approximate numerical linear dependency” occurs at large RKMax values and 2. Calculations become prohibitively expensive. RKMax should at most be between 9-10.

7.2 Ghostbands

Ghostbands are an inevitable consequence of investigating lithium with Wien2k. They manifest as an LAPW2 qtl error during the scf cycles, typically during the first cycle, but sometimes later as well. They are recorded the lapw2(_n).error files as:

```
'l2main' - QTL-B.GT.15., Ghostbands, check scf files
```

Ghostbands arise in a number of situations, each of which requires a different solution. Some of these are described below:

- **Muffin Tins badly chosen:** These appear when the muffin tins are too different, or if you ignore everything that **setrmt** does. **Solution:** Diagnose this by running the calculation with the **setrmt** values (might need to remove the core hole) or make the muffin tins more reasonable (might need to allow core leakage).
- **Local Orbitals need better initial guess:** Sometimes the starting points for local orbitals are not close enough to the converged values resulting in divergence. **Solutions:** Figure out which orbitals are causing the issue by looking in case.scf2(_n). The last lines of these files should have a line like this:


```
:WARN : QTL-B value eq. 99.95 in Band of energy -0.12674 ATOM=
2 L= 0
```

This message also tells us to look in case.in1(_st) and that the problem is with atom 2 and the L=0 orbital. In case.in1(_st), every independent atom is listed, eg:

WFFIL	EF= 0.50000	(WFFIL, WFPRI, EN
6.00	10	4 (R-MT*K-MAX; MAX L IN
0.30	1	(GLOBAL E-PARAMETER WI
0	0.30	0.000 CONT 1
0.30	1	(GLOBAL E-PARAMETER WI
0	0.30	0.000 CONT 1
0.30	1	(GLOBAL E-PARAMETER WI
0	0.30	0.000 CONT 1
0.30	1	(GLOBAL E-PARAMETER WI
0	0.30	0.000 CONT 1
0.30	1	(GLOBAL E-PARAMETER WI
0	0.30	0.000 CONT 1
0	1	(GLOBAL E-PARAMETER WI
0	0.30	0.000 CONT 1
0.30	3	(GLOBAL E-PARAMETER WI
0	-1.55	0.002 CONT 1
0	0.30	0.000 CONT 1
1	0.30	0.000 CONT 1
0	3	(GLOBAL E-PARAMETER WI
0	1.55	0.002 CONT 1
0	0.30	0.000 CONT 1
1	0.30	0.000 CONT 1
K-VECTORS FROM UNIT:4 -6.0 1.5 6		
Initial Guess		

L value for orbital

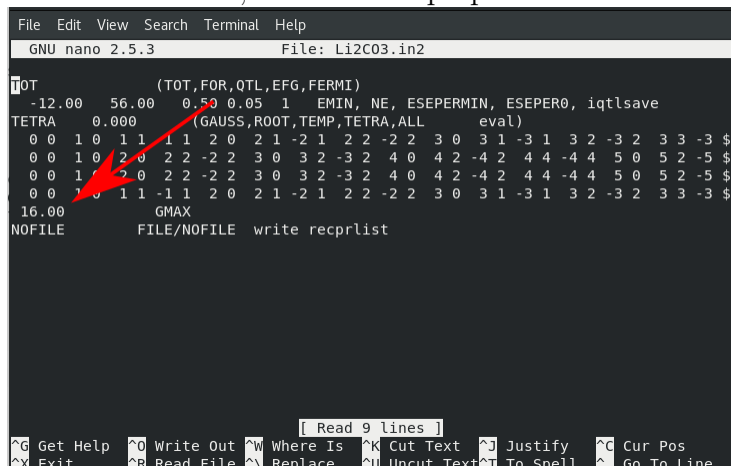
To handle ghostbands, either delete the relevant local orbital line (lines that match the atom and orbital number with energy guesses that are not 0.30) and reduce the number of orbitals for that atom correspondingly. A more refined option is to adjust the initial guesses to another value, and rerunning **run_lapw**.

7.3 NN in Optimization

Crashes the first scf cycle almost immediately, due to overlapping muffin tins resulting from a decreased cell size. Solution: decrease all muffin tin sizes before running “x optimize”.

7.4 GMax Value less than Gmin

Occurs in **dstart**, fix is to bump up the Gmax value in case.in2 from 12.00 to 14.00 or 16.00.



```
File Edit View Search Terminal Help
GNU nano 2.5.3 File: Li2C03.in2

TOT (TOT,FOR,OTL,EFG,FERMI)
-12.00 56.00 0.50 0.05 1 EMIN, NE, ESEPERMIN, ESEPER0, iqtlsave
TETRA 0.000 (GAUSS,ROOT,TEMP,TETRA,ALL eval)
0 0 1 0 1 1 1 1 2 0 2 1 -2 1 2 2 -2 2 3 0 3 1 -3 1 3 2 -3 2 3 3 -3 $
0 0 1 0 2 0 2 2 -2 2 3 0 3 2 -3 2 4 0 4 2 -4 2 4 4 -4 4 5 0 5 2 -5 $
0 0 1 0 2 0 2 2 -2 2 3 0 3 2 -3 2 4 0 4 2 -4 2 4 4 -4 4 5 0 5 2 -5 $
0 0 1 0 1 1 -1 1 2 0 2 1 -2 1 2 2 -2 2 3 0 3 1 -3 1 3 2 -3 2 3 3 -3 $
16.00 GMAX
NOFILE FILE/NOFILE write recprlist

[ Read 9 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

7.5 Ordering

If a calculation run apparently unchanged, a possible reason is that all the edits were overwritten by **init_lapw**. The order of a calculation should be:

- Make edits to case.struct file, ie. Muffin Tin Radius
- run **init_lapw**
- Edit .in files. eg. setting RKMax, dealing with ghostbands, inserting core hole, increasing k points...
- run **lapw**
- Increase Kpoints, run case specific calculations, and/or loop back to and run forwards from there (ie increasing k points/RKMax does not require rerunning **init_lapw**, but changing muffin tin radii does).
- **setrmt**
- **nn**
- **sgroup**
- **symmetry**
- **lstart**