**Disclaimer:** This document should *under no circumstances* be regarded as an authoritative reference for using VASP or iterpreting data obtained from VASP calculations. Its purpose is to provide – to the novice – sufficiently detailed introductions for setting up and performing simple calculations such that the reader can get their berrings using VASP. Many of the software's capabilities are not discussed herein. There are liable to be mistakes and/or errors, for which I apologize in advance, and request that you notify me of them so that this document can be revised and updated. I have liberally made use of the official online VASP manual, as well as the VASP wiki. Best of luck with your exploits; use at your own risk!

# 1 Setting up a calculation

Four input files are required to perform DFT calculations in VASP: `POSCAR`, `POTCAR`, `KPOINTS` and the `INCAR`

1. `POSCAR` contains the lattice coordinates and atomic positions. For electronic structure calculations of bulk solids (i.e. crystals, rather than slabs or surfaces) the `POSCAR` can be generated quite easily by exporting in .VASP format from VESTA. An example `POSCAR` is shown here to familiarize you with the formatting; note that the # and ! symbols can be used to add comments.

   ```
   PbO  # description of phase, usually chemical formula but can be anything
   1.0 # scaling parameter for lattice constants
   3.96 0 0  #lattice vectors in matrix form, in Angstroms!
   0 3.96 0
   0 0 5.01
   Pb O # atomic species present in POSCAR
   2  2 # number of atoms of each species in the UNIT CELL
   Direct # coordinates to be given in fractional, rather than cartesian, notation
   0.25 0.25 0.237 # 1st Pb position
   0.75 0.75 0.763 # 2nd Pb position
   0.75 0.25 0 # 1st O position
   0.25 0.75 0 # 2nd O position
   ```

   Note that the `POSCAR` contains *all of the atoms in the unit cell*, not just the atoms in the asymmetric unit. Although VASP does know crystallography and will recognize the symmetry of your input structure, it **does not** complete the crystal for you on the basis of the asymmetric unit the way that LMTO and the SPACEGROUP utility for ELK do.

2. `POTCAR` needs to contain the pseudopotentials for all of the atomic species that are involved in the calculation. Moreover, the `POTCAR` must contain the pseudopotentials **in the same order that the atomic species are listed** in the `POSCAR`. This is accomplished by catenating together a string of individual `POTCAR` files.

   Copy the `POTCAR` file for each of the elements contained within the structure you will be calculating into your working directory. The directory of potentials lives at `/usr/local/src/vasp/pot/potpaw_PBE/`; here you will find a collection of folders, inside which resides the `POTCAR` described by the name of the folder. For example, there are three folders for Pb: `Pb`, `Pb_d` and `Pb_d_GW`. `Pb` contains just 4 valence electrons, i.e. the core of the psuedopotential is [Xe] $4f^{14}$ $5d^{10}$. `Pb_d` contains 14 valence electrons, i.e. the core here is just [Xe] $4f^{14}$. Note that it is *significantly* more computationally expensive to use the

Pb_d pseudopotential than it is the 4-electron Pb potential. GW pseudo-potentials should not be used unless you *really* know what you're doing.[1]

So as to avoid problems associated with incorrectly ordering the potentials into the POTCAR file, and also because there are multiple pseudo-potentials available for most of the elements – all of which are labeled as just POTCAR in their respective "home" folders – I recommend using the following procedure to produce the POTCAR:

Using PbO as our running example:

(i) Copy the Pb potential to your working directory, appending the name POTCAR with the element: `$ cp /usr/local/src/vasp/pot/potpaw_PBE/Pb/POTCAR ./POTCAR_Pb`

Similarly, if you were planning to use the Pb_d pseudopotential then you would want to append the name accordingly: `$ cp /usr/local/src/vasp/pot/potpaw_PBE/Pb/POTCAR ./POTCAR_Pb_d`

(ii) Next copy the oxygen pseudopotential into your working directory: `$ cp /usr/local/src/vasp/pot/potpaw_PBE/O/POTCAR ./POTCAR_O`

(iii) Now string the potentials together into a new POTCAR file: `$ cat POTCAR_Pb POTCAR_O > POTCAR`

3. KPOINTS contains information about the $k$-point mesh. When calculating band structures this file will contain information about the high symmetry points of the first Brillouin zone between which you will calculate band dispersions.

   **When not performing band structure calculations** the format of the KPOINTS file will be similar to:

```
Automatic mesh
0 # number of k-points; when 0 --> automatic generation
Monkhorst-Pack # Alternately, Gamma centered meshes used often
6 6 4  # size of the mesh (6 x 6 x 4 points along reciprocal lattice vectors)
0 0 0 # optional shift of the k-mesh
```

   **For band structure calculations** the KPOINTS file would look something like this:

```
TET (tetragonal) G-X-M-G-Z-R-A-Z  X-R  M-A  # useful headder!
16   ! 16 grids  # 16 k-points between high symmetry points
Line-mode
reciprocal # k-points given in terms of reciprocal lattice
   0.000   0.000   0.000   ! \Gamma
   0.000   0.500   0.000   ! X
   0.000   0.500   0.000   ! X
   0.500   0.500   0.000   ! M
   0.500   0.500   0.000   ! M
   0.000   0.000   0.000   ! \Gamma
   0.000   0.000   0.000   ! \Gamma
   0.000   0.000   0.500   ! Z
   0.000   0.000   0.500   ! Z
```

[1] $GW$ potentials are used for quantitative calculations of excitation energies employing the dynamically screened interaction, also known as the $GW$ approximation, in which the self-energy is the product of the single-particle Green function, $G$, and the screened interaction, $W$.

```
0.000   0.500   0.500   ! R
0.000   0.500   0.500   ! R
0.500   0.500   0.500   ! A
0.500   0.500   0.500   ! A
0.000   0.000   0.500   ! Z
0.000   0.500   0.000   ! X
0.000   0.500   0.500   ! R
0.500   0.500   0.000   ! M
0.500   0.500   0.500   ! A
```

Note that the high symmetry $k$-points are given in pairs, i.e. to calculate the $\Gamma$ to X to M etc. dispersions, you must list the points corresponding to $\Gamma$, X then X, M, then M and so forth. (This is distinct from ELK in which the intersections are inferred.)

4. INCAR tells VASP what to calculate and specifies parameters for the calculation.

   There are *lots* of parameters that *can* be included in the INCAR file, however, **it is always better to include only those parameters that you are familiar with and care about**. In fact, simple calculations (electronic relaxation to ground state) can be performed with an empty INCAR file, as long as you make sure to create a blank file named INCAR – in this case VASP will simply apply all of the default values to the calculation.

   I recommend at the following parameters for even the most basic INCAR, many of which facilitate good housekeeping:

   `SYSTEM = chemical formula` (does not influence calculation, but useful housekeeping)

   `ISTART = 0, 1, or 2` type of job to be performed

   `ISTART = 0` : start from scratch

   `ISTART = 1` : Restart job, orbitals are read from previously written WAVECAR, but the *plane waves are redefined*

   `ISTART = 2` : Restart job, orbitals are read from previously written WAVECAR, but the *plane waves are unchanged*

   NOTE: if cell volume and/or shape is unchanged, then `ISTART = 1` and `ISTART = 2` will yield the same result.

   `PREC = Low, Medium, Normal,` or `Accurate` ; in essence determines the precision of calculations by inlfuencing the default settings of `ENCUT, NGX(Y,Z), NGX(Y,Z)F,` and `ROPT`.

   For most purposes the default setting `PREC = Normal` is likely to be sufficient. `PREC = Accurate` is recommended when accurate forces are needed (e.g. phonons). There are other settings available for PREC not discussed here, but that should be used when certain exchange-correlation functionals are used.

   Note that stating a specific ENCUT in the INCAR file will override the selection made by PREC.

   ROPT is used for real-space projectors and controls the number of grid points within the integration sphere around each atom.

   `NBANDS =` determines the *actual* number of bands included in your calculation.

   Recommended setting: `NBANDS =` $\frac{\#\text{ of electrons}}{2} + \frac{\#\text{ of ions}}{2}$

   For large systems: `NBANDS =` $\frac{\#\text{ of electrons}}{2} + \frac{\#\text{ of ions}}{4}$ probably OK.

   **But for transition metals, use up to** `NBANDS =` $\frac{\#\text{ of electrons}}{2} + (2 \times \#\text{ of ions})$

`ENCUT =` cutoff energy for plane-wave basis sets – controls the completeenss of the basis set. This is an important parameter that should usually be set manually. The default value of `ENCUT` is the largest `ENMAX` found in the `POTCAR` file, which might be sufficient, although larger values are often required and convergence with respect to `ENCUT` should be checked. A very reasonable – if perhaps excessive – starting value is `ENCUT = 520` (eV). You may even find that you rarely have to increase `ENCUT` beyond 520 eV.

`LREAL = .False.` or `Auto` or `On`. Determines whether projection operators are evaluated in reciprocal space (`LREAL = .False.`, the default) or real space (`Auto` or `On`).

The use of `LREAL = Auto` is preferred to `LREAL = On`, and should be set as such when your system contains more than 20 atoms. Beware that real-space optimization always results in small but significant errors; however, these are really only a problem if you are interested in energy differences on the order of a few meV.

`ISMEAR =` determines how partial occupancies are set for each orbital. Only the commonly used values of `ISMEAR` are detailed here:

`ISMEAR = 1` : Default setting, Methfessel-Paxton scheme. `ISMEAR = N` where $N > 0$ actually indicates the order $N$ of the Methfessel-Paxton smearing scheme. **For semiconductors and insulators do not use** `ISMEAR` $> 0$. However, **for relaxation in metals always use** `ISMEAR = 1` or `ISMEAR = 2`.

`ISMEAR = 0` : Gaussian smearing – gives smooth DOS curves! Note however that for semiconductors and insulators the Fermi energy is referenced to the center of the gap, rather than the top of the valence band.

`ISMEAR = -5` : Tetrahedron method (with Blöchl corrections); **make sure to use a $\Gamma$ centered $k$-mesh!**. *This is the preferred smearing setting for relaxations in semiconductors and insulators*. **This is also the preferred setting for all DOS calculations (including metals; static run, no relaxation) and very accurate total energy calculations**. Using `ISMEAR = -5` for calculating the DOS references the Fermi energy to the top of the valence band. However, for large cells or if you find your DOS to be contain unusually "peaky" features, try `ISMEAR = 0` with a small `SIGMA` (ca. 0.05 eV).

`SIGMA =` determines the width of smearing, in eV

Default is `SIGMA = 0.2`, which is a very reasonble value for metals.

Obviously the number of parameters can easily become overwhelming. Below is an example of how I like to set up my `INCAR` file. Note that many parameters have either been commented out, or contain comments after their values that help me keep track of what I'm asking the program to do. Additionally, I like to divide the `INCAR` into sections that describe the processes the parameters influence – the organziation is loosely consistent with how VASP groups flags in the `OUTCAR` (the file containing detailed information about the results of your calculation). You will also notice that there are parameters contained here that I have not included in the list above. At the end of this file you will find a "crib sheet," a reference guide to the parameters that I find myself using most frequently. This guide is *by no means all-inclusive*. In fact, there is extensive documentation online (VASP), which I encourage you to browse. In my experience it is easy to get lost in the sheer vastness of the online VASP resources when you are starting from scratch; hopefully, however, this document will provide a sufficient introduction and foundation from which to start digging into this pervasive and powerful plane-wave code more deeply.

Ok, on to the example INCAR:

```
SYSTEM = PbO, static

########## start parameters
ISTART = 0
PREC = Normal
NBANDS = 40
#
##ICHARG = # set to 11 for DOS or BAND calculations (non-self consistent)
##ISPIN =  # 1 for non-spin-polarized (Default), 2 for spin-polarized
##LSORBIT = # .False. (default), .True. to turn on spin-orbit coupling
##LNONCOLLINEAR = # .False. (default), .True. for non-collinear magnetic materials
##LASPH = # .False. (default); .True. when 3d elements present (and other situations)

########## electronic relaxation parameters
ENCUT = 450
LREAL = .False.
#
##ALGO = # minimization algorithm

########## DOS values
ISMEAR = -5
SIGMA = 0.2
#
##LORBIT =  # set to 10 or 11 for (non-self-consistent) DOS calc
##EMIN = -10 # minimum energy of DOS curve in eV, default = -10
##EMAX = 10 # max energy of DOS curve in eV, default = 10
##NEDOS = 5000  # number of points in DOS curve
```

As you can see, the majority of parameters have been commented out – the default parameters are well suited to the calculation (a static self-consistent electronic minimization "from scratch"). It is my preference, however, to include flags (appropriately commented out) that I am likely to use in subsequent calculations.

As a final note about setting up the INCAR, the user should be aware that the default exchange correlation functional is the Generalized Gradient Approximation of Perdew, Burke and Ernzerhof (GGA-PBE). The xc functional can be changed as desired (e.g. GGA+$U$, HSE06, etc.) and will be discussed later in this document, though GGA-PBE is appropriate for many calculations.

You are now ready to run a calculation! This can be accomplished by simply running the command `vasp` in the terminal (from the appropriate working directory, of course).

Note however that a preferred method for starting a calculation is to execute the following command: `$ nohup vasp >& vasp.log &` ; this will run VASP in the background and copy all of the information that would appear directly in your terminal window had you simply run `$ vasp` into a file called `vasp.log`. (The log file can be named anything you want, so long as it does not coincide with

any of VASP's output file names.) I recommend making an `alias` in your `.bashrc` file, for example: `alias runvasp='nohup vasp >& vasp.log &'`

# 2 Calculating the DOS

1. Run a static self-consistent calculation (`ISTART = 0` and `NSW = 0`) with a reasonable (but not especially large number) of $k$-points. This will produce a sufficiently accurate charge density, stored in the `CHGCAR`, to be used in the following step.

   It is recommended to run the self-consistent calculation using the same type of smearing that you will use for the non-self-consistent calculation of the DOS.[2] The most accurate smearing method is `ISMEAR = -5` but you are not obliged to use this (tetrahedral method) as it sometimes leads to "ugly" DOS. **If you DO use** `ISMEAR = -5` **you MUST use a** $\Gamma$ **centered** $k$**-mesh**:

   ```
   Automatic mesh
   0
   Gamma
   4  4  4
   0. 0. 0.
   ```

2. In the `INCAR` file add `ICHARG = 11` and either `LORBIT = 10` or `LORBIT = 11`. These additional flags tell VASP that you want to use the previously calculated charge density (non-self-consistent calculation) and that you want to output a `PROCAR` file (in addition to the `DOSCAR`), respectively. The `PROCAR` file contains the site and $l$-projected (`LORBIT = 10`) or $lm$-projected (`LORBIT = 11`) density of states.

3. Modify the `KPOINTS` file and significantly increase the number of points along each reciprocal lattice vector.

4. Run the calculation

5. Run the `split_dos` script. This will produce $N + 1$ new files, where $N$ = number of atoms in the unit cell, called `DOS0`, `DOS1`, etc. `DOS0` contains the total DOS (as well as the integrated DOS in the third column of the file); `DOS1` to `DOS`$N$ contain the orbital projected DOS for each atom in your `POSCAR` file (in order of appearance). The number of columns in the `DOS`$N$>0 files depends on whether you chose `LORBIT` of 10 or 11, but both start with the energy ($x$); for $l$-projected DOS (`LORBIT = 10`) there will be 3 columns of $y$ values (4 in the case of elements containing $f$ electrons) for the summed $s$, $p$ and $d$ contributions. If `LORBIT = 11` then the $y$ columns correspond to the single $s$, three $p$, and five $d$ orbitals, respectively. These files can all easily be plotted using XMGRACE (e.g. `$ xmgrace DOS0 -nxy DOS1 -nxy DOS2`).

# 3 Calculating the band structure

It is important to keep in mind that band structure calculations should *almost* always be conducted using the primitive unit cell. The reason for this is two-fold: (1) doing so reduces the computation time. (2) We

[2] There are situations when this "rule" is violated, such as when DOS calculations are performed using hybrid functionals to describe the exchange-correlation.

are only interested in dispersions within the first Brillouin zone – anything beyond the first zone is redundant.

With that said, there are occationally circumstances where band structure calculations using the conventional cell are desireable, but outside of such circumstances doing so should be avoided.

For those of you familiar with the LMTO and/or ELK codes, you are likely aware that the conventional unit cell input via `lminit.run` or with the aid of the `spacegroup` utility is either automatically or easily, respectively, converted into the primitive cell. VASP **does not do this!** However, there is an incredibly handy online utility for manipulating and generating input files for VASP. **In particular, this utility is incredibly useful when performing band structure calculations as it allows one to very easily convert to the standard primitive unit cell and to generate the corresponding list of special $k$-points between which band dispersions will be calculated.** The development of ACONVASP ONLINE was spearheaded by Stefano Curtarolo, in collaboration with numerous other people. If you publish work that has benefit form the use of ACONVASP – and considering how much easier it makes life I strongly encourage you to use it – please make sure to cite the authors:
S. Curtarolo, W. Setyawan, G. L. W. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M. Mehl, H. T. Stokes, D. O. Demchenko, and D. Morgan, AFLOW: an automatic framework for high-throughput materials discovery, *Comp. Mat. Sci.* **58**, 218 (2012). [doi]

**However... (and THIS IS VERY IMPORTANT TO BE AWARE OF):** The standard primitive cell and corresponding Brillouin zones output by ACONVASP are in certain instances more rigorous than those often encountered in the literature and/or defined by other programs such as LMTO. A portion of the relevant passage from the article is reproduced here (footnote).[3]

So, to perform a band structure calculation:

1. Assuming you are starting from a structure described by the conventional unit cell, open the structure with VESTA and export in .VASP format. (No need to do this if you have already constructed a `POSCAR` file)

2. Go to ACONVASP ONLINE, copy the contents of your `compound.vasp` (or `POSCAR`) file into the box that says "`Input POSCAR here:`", select "Kpath in the reciprocal space for band structure calculations *with respect to the reciprocal vectors generated from the standard primitive lattice vectors*" in the **Symmetry** section, and click submit (the output will open in a new tab).

3. Copy the standard primitive setting-converted structure appearing at the top of the ACONVASP output into your `POSCAR`. (Note that your conventional cell may have already been the primitive cell).

4. Copy the $k$-points information appearing below "`// KPOINTS TO RUN ******`" into a `KPOINTS_bnds` file (refer to description of `KPOINTS` files in section 1 of this document for example of what this would look like). You will probably wish to increase the number of points

[3]Notably for triclinic, monoclinic, and rhombohedral systems, the shape of Wigner-Seitz cell of the reciprocal lattice depends non-trivially on the lattice vectors. For this reason, some researchers use the parallelepiped of primitive reciprocal lattice vectors, centered at $k = 0$, to define a plausible BZ. Even though the energy is continuous throughout such BZ, its faces, in general, are not parallel to the symmetry planes of the lattice. Therefore, there is no guarantee that any line connecting two k-points on a BZ face will be a symmetry line. In addition, since parallelepiped unit cells have only eight points, one at each corner of the IRBZ, one would miss some symmetry points on the Wigner-Seitz cell of reciprocal lattice. Consequently, a complete irreducible set of symmetry lines would not be obtained.

Also note the lovely illustration of your compound's Brillouin zone that is generated by ACONVASP and displayed at the bottom of the page!

5. Run a static self-consistent calculation (`ISTART = 0` and `NSW = 0`) with a reasonable (but not especially large number) of $k$-points using an automatically generated $k$-point mesh in the `KPOINTS` file. This will produce a sufficiently accurate charge density, stored in the `CHGCAR`, to be used in the following non-self-consistent calculation. For semiconductors and insulators use `ISMEAR = 0`; for metals use `ISMEAR = 1`.

6. Replace the `KPOINTS` file with `KPOINTS_bnds` (`$ cp KPOINTS_bnds KPOINTS` or `$ mv KPOINTS_bnds KPOINTS`).

7. Modify the `INCAR` to include the parameter `ICHARG=11` and run the calculation. (Theoretically you can make a fat band calculation by also including `LORBIT = 11` in your `INCAR`, although constructing a plot of the fat bands is non-trivial.)

8. Lastly you will want to plot the band structure. Unfortunately doing so is not nearly as straightforward as it should be. I am only aware of one quasi-convenient way to export the band structure data in a format that can be read by XMGRACE: via P4VASP, a rather buggy application that is capable of reading the `vasprun.xml` file.

   The most painless way I have found to get band structure data out of P4V is to export as both a `.agr` and `.dat`. While the exported XMGRACE file is fine for cursory inspection, it requires extensive reformatting to produce a publication quality graphic. Rather than reformatting, I prefer to import the `.dat` file (which only contains the bands themselves, does not include the vertical lines denoting $x$ coordinates of the high symmetry $k$-points) into XMGRACE, and to then get the positions of the high symmetry points by opening the `.agr` file in a text editor. In particular if your band structure contains any discontinuities in the dispersions – for example, if your `KPOINTS` file has Γ-X-blah-blah-A-Z X-R M-A – then I find it to be significantly easier to make a plot from scratch.

*A note about error messages one might encounter when using the standard primitive unit cell*, which can sometimes confuse VASP. Two of the most common error messages I have noticed are (1)

```
VERY BAD NEWS! internal error in subroutine IBZKPT:
Reciprocal lattice and k-lattice belong to different class of lattices.
Often results are still useful...
```

and (2)

```
VERY BAD NEWS! internal error in subroutine SGRCON:
Found some non-integer element in rotation matrix.
```

(1) This is typically observed during the calculation of the charge density using an automatically generated $k$-mesh (either Monkhorst-Pack or Gamma centered). It is my *non-expert opinion* that this first error is sometimes encountered even when one has seemingly chosen an appropriate $k$-mesh (i.e. appropriate dimensions). As the message indicates, however, as long as the error does not crash VASP the results might be fine. One thing to try is switching from an even numbered $k$-mesh to an odd numbered $k$-mesh (e.g. from $6 \times 6 \times 4$ to $5 \times 5 \times 3$).

(2) I have no idea what this message really means, and the VASP discussion-board administrators discuss it somewhat cryptically. They do, however, offer a work-around: set `ISYM = 0` in the `INCAR`.

### 3.1 Sampling the same Brillouin zone defined by LMTO

While ACONVASP does an excellent job of rigorously defining Brillouin zones and providing lists of high symmetry $k$-points between which to calculate dispersions, for certain crystal systems these will be notably different than the BZs and corresponding dispersions that are defined and calculated in LMTO. In fact, most of the band structures one comes across in the literature correspond to the "standard" BZ definitions used in LMTO. For the sake of comparison between DFT codes and also for parity with previously published work...:

1. Make a `POSCAR` using the lattice matrix and atomic positions generated by LMTO, which appear in the `CTRL` file.

   Atomic positions in `CTRL` are given in **cartesian** coordinates. This must be accordingly specified in the `POSCAR` (i.e. where usually `Direct` appears, replace with `Cartesian`). Recall that the lattice vectors in the `CTRL` file are given in units of Bohr! – as long as you convert the lattice scalar `ALAT` to angstroms there is no need to modify values in the lattice matrix.

2. Make a `KPOINTS` file using the high symmetry points specified in the `SYML` section of `CTRL`. As with the atomic positions, the points generated by LMTO are in **cartesian** coordinates. Thus, your `KPOINTS` file will look something like:

```
LMTO-setting C-cent mono grid
20
Line-mode
Cartesian
0.169445 0.500000 0.000000 ! V
0.000000 1.000000 0.000000 ! Z
0.000000 1.000000 0.000000 ! Z
0.000000 0.000000 0.000000 ! G
0.000000 0.000000 0.000000 ! G
0.064225 0.000000 0.134409 ! A
0.064225 0.000000 0.134409 ! A
0.064225 1.000000 0.134409 ! M
0.064225 1.000000 0.134409 ! M
0.233670 0.500000 0.134409 ! L
```

# 4 Spin polarized calculations

## 4.1 Ferromagnetic compounds

1. In the `INCAR` set `INSPIN = 2` to turn on spin-polarization, and add the `MAGMOM` flag. `MAGMOM` describes the magnetic moment of each atom in the `POSCAR` entered sequentially. For example, if there are 4 atoms in the `POSCAR` and the first two are magnetic, the flag would look like this:

```
MAGMOM = 1.5 1.5 0 0
```

   A good rule of thumb for assigning the initial moments is to use 1.2–1.5 times the experimental moment.

It is also helpful to set the LORBIT flag (LORBIT = 1, 2, 10, or 11), which prints $spd$-decomposed moments into the OUTCAR (look for magnetization (x)).

2. Run the calculation and check mag = in the .log file.

   Voila.

**General tip for magnetic calculations:** If you are having trouble converging to the desired configuration, it can help to start from the non-magnetic groundstate. In this case:

1. Run the calculation with ISPIN = 1
2. In the INCAR, change to ISPIN = 2 and ICHARG = 1.
3. Run the calculation again; the non-magnetic WAVECAR and CHGCAR will be used as a starting point.

## 4.2 Antiferromagnetic compounds

VASP determines whether the magnetic moments supplied in the MAGMOM flag break symmetry; thus, many antiferromagnetic calculations can quite easily be performed *without constructing a super-cell* by specifying anti-parallel moments:

1. For example, an antiferromagnetic metal containing two atoms in the unit cell (such as Cr) could be calculated as such by including the following flag:

   MAGMOM = 1 -1

## 4.3 Non-collinear spin arrangements

Complex spin arrangements such as spirals can be modeled in VASP, although doing so is beyond the scope of this guide (and beyond the expertise of the guide's author). Please refer to the next section on using spin-orbit coupling to get a broad sense for the type of setup required when the LNONCOLLINEAR flag is turned on. Note that the non-collinear flag is automatically turned on when spin-orbit coupling is used, but that you are not required to use spin-orbit coupling when performing a non-collinear calculation.

## 4.4 Magnetic calculations with spin-orbit coupling

Spin-orbit coupling is turned "on" by simply setting the flag LSORBIT = .True. ; doing so automatically sets the flag LNONCOLLINEAR = .True.. Thus, magnetism – and more specifically non-collinear magnetism – is assumed automatically if the spin-orbit interaction is included. (You can, of course, start with zero-valued moments)

Since the concept of up vs. down spin has no meaning for noncollinear magnetism or spin-orbit calculations, the ISPIN = 2 flag also has no meaning! Fortuantely, VASP recognizes this and ignores it when spin-orbit coupling is included.

Link to Fully unconstrained noncollinear magetism within the PAW method

### 4.4.1 General setup of SOC-enabled calculations

1. By including the spin-orbit interaction, the spin is inherently coupled to the crystal structure – therefore a spin quantization axis (SAXIS) must be defined in the INCAR:

   By default, SAXIS = 0+ 0 1, where 0+ indicates an infinitesimally small positive number in the $x$ direction.

   By defining the spin quantization axis, all magnetic moments written and read by VASP are evaluated **with respect to this axis**.

   link to VASP wiki on SAXIS

2. Now the components of the moments specified in the MAGMOM flag must be done so in three dimensions (i.e. given as $x$,$y$,$z$ vectors).

   Since the directions of the local moments are defined relative to the global quantization axis, the recommended way to initialize calculations with magnetic moments parallel to a chosen vector is to use:

   ```
   MAGMOM = 0 0 moment  ! where "moment" is an integer!
   SAXIS =  x y z  ! where x y and z are integers!
   ```

   Note that three numbers must be specified in the MAGMOM flag for each atom in the POSCAR

3. Double the number of bands, i.e.:

   ```
   NBANDS =  ! two times the number of bands used for the collinear calculation
   ```

### 4.4.2 Calculating magnetic anisotropies

1. If you have $d$-elements present set LMAXMIX = 4; if $f$-elements are present set LMAXMIX = 6. **Perform a self-consistent collinear calculation** (i.e. ISPIN = 2 and LSORBIT = .False., etc.) so as to obtain a CHGCAR to be used in subsequent (non-selfconsistent) calculations.

2. Add the follow tags to the INCAR and set as is appropriate for your desired calculation:

   ```
   ICHARG = 11  ! non-selfconsistent calc
   LSORBIT = .True.
   SAXIS =  ! direction of the magnetic field
   MAGMOM =  ! make sure that the moments are now specified as vectors!
   ```

3. Run the calculation. VASP aligns the spin quantization axis parallel to SAXIS – thereby making the magnetic field parallel to SAXIS. By comparing the energies obtained for different orientations of SAXIS the magnetic anisotropy can be determined.

# 5 DFT+$U$ Calculations

The type of DFT+$U$ calculation performed by VASP – that is, LDA+$U$ or GGA+$U$ – depends on whether you are using LDA or GGA pseudopotentials. Note that the same flags (LDAU=, LDAUTYPE=, etc.) are used in both cases.

1. Add the following flags to the INCAR:

   ```
   LDAU = .True.
   LDAUTYPE =
   LDAUL =
   LDAUU =
   LDAUJ =
   ```

2. There are three possible settings for LDAUTYPE: 1, 2, or 4.

   LDAUTYPE = 1 : Rotationally invariant LSDA+$U$ (or GGA+$U$) as implemented by Liechtenstien *et al.* In this case the on-site Coulomb and exchange parameters, $U$ and $J$, are independent.

   LDAUTYPE = 2 : A simplified rotationally invariant LSDA/GGA+$U$ approach described by Duvarev *et al.* In this case $U$ and $J$ are not treated independently, only the difference $U - J$ is used.

   LDAUTYPE = 4 : Takes the same form as type 1, but in the absence of spin-density exchange splitting... i.e. this is used for performing LDA+$U$ calculations, rather than LSDA+$U$. Presumably this setting has no meaning when GGA potentials are used.

3. LDAUL = : A list specifying the $l$-quantum number for each species in the POSCAR, sequentially, for which the on-site interaction is added.

   LDAUL = -1 : no on-site terms added

   LDAUL = 1 : $p$-electrons

   LDAUL = 2 : $d$-electrons

   LDAUL = 3 : $f$-electrons

   Example: For MnBi, which contains 2 Mn and 2 Bi atoms in the unit cell (in that order), LDAUL = 2 -1.

4. LDAUU = : list of effective on-site Coulomb interaction parameters for each species in the POSCAR.

   For MnBi, the flag would look something like this: LDAUU = 5.1 0. (Note that the value $U = 5.1$ eV is probably too large!)

5. LDAUJ = : list of effective on-site exchange interaction parametrs for each species in the POSCAR

   For MnBi, the flag would look something like this: LDAUJ = 0.1 0.

**If you are performing a band structure calcultion using DFT+$U$**: you will need to add an additional flag, LMAXMIX.

For +$U$ calculations with $d$-elements, set LMAXMIX = 4
For +$U$ calculations with $f$-elements, set LMAXMIX = 6

# 6 Calculating the ELF

*Theoretically* it is very easy to calculate the ELF with VASP: add the flag LELF = .True. to the INCAR and change to PREC = Accurate. This will output an ELFCAR that can be directly read into VESTA.

Note, however, that despite the ease of accessibility and pervasive presence of VASP-calculated ELFs in the literature, caution is advised when using this feature; closely inspect the generated ELFs!

Another option, as an alternative or complement to the ELF, is the generation of Wannier functions. These offer a representation of the shape of molecular orbitals, enabling the identification of bonds and lone pairs. The WANNIER90 code interfaces with VASP, but as I have not yet made any attempts to "play" with it, it is beyond the present scope of this document.

# 7 Calculating COHPs (and COOPs!) via LOBSTER

A couple things to keep in mind: (1) do not use ultrasoft pseudopotentials, (2) `_sv` potentials are to be avoided. More importantly: the COHP algorithm implemented in LOBSTER does not implement $k$-point symmetry – i.e. the entire $k$-point mesh must be used, rather than just the irreducible points. This significantly increases the computational time and cost... Thus, it may not be possible to calculate COHPs with as fine of a $k$-mesh as used for DOS calculations.

**Very important**: the setting `NBANDS` is critical to the success of a LOBSTER calculation. The number of bands must be set as the number of **BASIS FUNCTIONS** – i.e. *occupied* orbitals – that will be used by LOBSTER. Let's look at a couple quick examples:

The standard `paw_PBE POTCAR` for Ti contains 4 electrons; the unit cell of titanium metal contains 2 Ti atoms. Here we would want to use `NBANDS = 18`, which comes from including 9 bands for each Ti atom ($1\times 4s + 3\times 3p + 5\times 3d$). Note that this is liable to be significantly more bands than generally recommended for a generic VASP calculation. Note also that the number of bands **must be exactly equal to the number of basis functions**.

PbO contains 2Pb and 2O atoms in the unit cell. Assuming the basis functions of interest are the $s$ and $p$ orbitals of both elements, then `NBANDS = 16` would be used.

Ok, so here's how it's done:

1. First, a suitable `KPOINTS` file containing the entire $k$-mesh must be generated. To do this, set up your static calculation as you normally would – with a generic `KPOINTS` file, but adding the following flags to the `INCAR`:

   ```
   ISYM = 0
   LSORBIT = .True.
   ```

2. Now start the calculation... But, as soon as the `IBZKPT` file has been generated you should stop the calculation.

3. Replace the `KPOINTS` file with the `IBZKPT` file: `$ cp IBZKPT KPOINTS`

4. Now turn off (i.e. comment out) the `ISYM` and `LSORBIT` flags.

5. Run the complete static self-consistent calculation, which will generate the `WAVECAR` LOBSTER needs to work its proverbial magic.

6. Make a new file `lobsterin`, which will be read by LOBSTER. An example file is shown below; note that there are three options for chosing which COHPs to calculate – comment out the flags you do not wish to use (none of the flags that determine which COHPs are generated have been commented out in this example).

```
! comments preceeded by !
!
! Energetic window in eV (relative to the Fermi level):
COHPstartEnergy -10
COHPendEnergy 5
!
! specify valence orbitals to use:
includeorbitals s p d
!
! OPTION 1:
! Define the pairs for which COHP analysis should be done;
! atoms are numbered as per order in the POSCAR file.
cohpbetween atom 1 atom 10
!
! OPTION 2:
! Orbital resolved COHPs: this will give you all the
! s-s, s-p_x, ..., p_z-p_z etc COHPs for the specified atom pairs.
cohpbetween atom 1 atom 2 orbitalwise
!
! OPTION 3:
! Generate the COHP pairs automatically:
! inlcude all pairs in a given distance range (in Angstrom)
cohpGenerator from 1.4 to 3.0
!
! Lobster chooses the type of basis set automatically for you.
! If you wish to override this selection for some reason, you can do
! so by uncommenting one of the following lines:
! basisSet bunge ! works up to Xe (Z<55)
! basisSet koga ! works up to Lr (Z<104)
```

7. Run lobster! `$ lobster-1.0.1`

   Note that there are two versions, `lobster-1.0.0` and `lobster-1.0.1`. It would be prudent to run the calculation using both versions to check for parity (best accomplished in a separate folder!).

   Lobster outputs the following files:

   `lobsterout` : log file

   `DOSCAR.lobster` : orbital projected DOS. This file contains $N + 1$ sets of data, where $N =$ number of atoms in the `POSCAR`. Note that the output does not separate these sets with carriage returns; to read this into XMGRACE you should edit the file: comment out the first 6 lines, comment the header that separates each set, and add a carriage return between sets so that they are not read as a single continuous data set.

   `COHPCAR.lobster` : projected COHPs requested in `lobsterin`. Column 1: energy ; Column 2: COHP averaged over *all specified atom pairs*. Column 3: integrated COHP averaged over *all specified atom pairs*. Column 4: COHP of 1st interaction. Column 5: integrated COHP of 1st interaction. Column 6: COHP of 2nd interaction. etc. etc.

`COOPCAR.lobster` : projected COOPs; analogous format to `COHPCAR.lobster`

`ICOHPLIST.lobster` : list of integrated (up to the Fermi level) projected COHP values.

`ICOOPLIST.lobster` : list of integrated projected COOP values.

# 8 Bader charge analysis

Assuming you have already run a static self-consistent calculation, the Bader charges can quite easily be calculated:

1. In the `INCAR`, change to `ISTART = 1` and including the following:

   ```
   ######### Bader parameters
   LCHARG = T
   LAECHG = T
   LADDGRID = T
   NGXF =
   NGYF =
   NGZF =
   ```

   Starting values of `NG(X,Y,Z)F` should be chosen as multiples of the default values listed in the `OUTCAR`. For a first go-round multiply each by a factor of 2.

2. Run the calculation; note that three new files are created: `AECCAR0` (core charges), `AECCAR1` and `AECCAR2` (valence charges).

3. Run `$ chgsum.pl AECCAR0 AECCAR2`. This will create the `CHGCAR_sum` file.

4. Run `$ bader CHGCAR -ref CHGCAR_sum`. The charges are written to `ACF.dat` and `BCF.dat`.

5. To obtain the *Bader charge* for an atom, subtract the charge shown in `ACF.dat` from the number of valence electrons specified in the atom's pseudo-potentials. For example, if the charge given in `ACF.dat` is 6.5, and the pseudopotential for that atom has 7 electrons, the Bader charge is 0.5.

6. Now repeat the calculation with increasing values for `NG(X,Y,Z)F` until the Bader charges converge.

# 9 DOS / band-structure calculations enlisting the HSE06 hybrid functional

Very important: **DOS and band structure calculations using HSE CANNOT be continued from an exixting CHGCAR file (i.e. NEVER set** `ICHARG = 11`**)**: the non-local exchange is not determined by the charge density but rather by the density matrix and/or the KS-orbitals

## 9.1 HSE DOS

Note: HSE calculations can be *very* computationally intensive. Whereas for GGA/LDA calculations the $k$-points can be increased dramatically for accurate DOS calculations since the charge densities are held constant, such exceptionally fine $k$-point meshes are neither pratical nor necessesary for HSE calculations.

1. Run a static self-consistent calculation with the following flags included in the INCAR:

```
########## HF-parameters HSE06
LHFCALC = TRUE
HFSCREEN = 0.207
AEXX = 0.25
TIME = 0.4
PRECFOCK = Normal
ALGO = Normal
```

2. Run the calculation

In theory it is that simple, but in practice one might encounter a variety of problems...

## 9.2 HSE bands

HSE band structure calculations require the use of a little trickery...

1. Run a self-consistent HSE calculation with a conventional KPOINTS file.
2. Copy the IBZKPT file to the KPOINTS file (`$ cp IBZKPT KPOINTS`)
3. To the end of the KPOINTS file add all of the desired $k$-points along high-symmetry lines of the Brillouin zone and set their weight to zero; note that you must include iterpolated $k$-points between high symmetry points of the BZ. For example:

```
.
.
.
    0.25000000000000    0.50000000000000    0.50000000000000              2
    0.50000000000000    0.50000000000000    0.50000000000000              1
0 0 0.5 0
0 0 0.45 0
0 0 0.4 0
0 0 0.35 0
0 0 0.3 0
0 0 0.25 0
0 0 0.2 0
0 0 0.15 0
0 0 0.1 0
0 0 0.05 0
0 0 0 0
.
.
.
```

   Make sure to change the total number of $k$-points (the second line of the KPOINTS file) to reflect the additional points you add.

4. Add `NELMIN = 5` to the `INCAR` and run the calculation.
5. The band structure can be visualized using P4V.

A tip: you might only want to calculate an HSE band structure along high symmetry lines of interest.

# 10 Structure relaxation

One of the many excellent features in VASP is the ability to perform relaxations (optimizations). Here we will only consider routines used for conducting ionic relaxations of bulk materials, but one should be aware that relaxations can also be (and commonly are) performed for slabs (surfaces), clusters and nanoparticles, molecules, etc.

In general, there are two ways to perform structural relaxations: (1) by allowing VASP to optimize the cell shape (lattice symmetry), cell volume (lattice constants), and ionic positions until a minimum is reached (`ISIF = 3`); and (2) performing a series of calculations at a variety of *fixed cell volumes* (`ISIF = 4`) and then fitting the obtained energies to the Murnaghan Equation of State to obtain the "optimal" cell volume.

In my experience, most of the hardcore DFT people who do truly exceptional work opt for relaxation strategy (2), but that is not to say strategy (1) lacks utility for us experimentalists.

## 10.1 Volume-unconstrained structural relaxations

1. Specify [at least] the following parameters in the `INCAR`:

   `ENCUT =` `#` *at least* $1.3\times$ the largest `ENMAX` value contained in the `POTCAR` file. For example, `ENMAX` for oxygen is 400 eV – so if this is the largest `ENMAX` value of any atoms contained in the structure then you would want to choose `ENCUT` of at least 520 eV.

   ```
   ISMEAR = 0   # for a semiconductor or insulator, 1 for metals

   ########## ionic relaxation parameters
   NSW = 60  # max steps for optimization ; NSW = 0 (default) for static calc
   IBRION = 1  # set to 1 when close to minima, 2 for difficult relaxations
   ISIF = 3 # determines type of relaxation performed (3 --> ions, shape, and vol)
   EDIFFG = -0.01  # convergence criterion: forces on all atoms < 0.01 eV/Angst
   NELMIN = 4 # minimum number of electronic steps, 4-8 for ionic relaxations
   ```

   **An important note about** `EDIFFG`: for most relaxations the use of a negative value, which tells VASP to use the forces on the ions as the criterion for convergence, is a good choice. *However, if the compound of interest only contains atoms on special (Wyckoff) positions in the unit cell, then the force criterion makes little sense since the atoms are not likely to relax away from their sites (unless you start from a very bad structure, I suppose...).* **In this case use a positive value for** `EDIFFG`, which sets the convergence criterion for ionic relaxation to be when the change in total energy between ionic steps is less than the provided value. The default value of `EDIFFG` may very well suffice, where `EDIFFG` $= 10\times$ `EDIFF`. (`EDIFF` is the convergence criterion for electronic relaxation)

2. Run the calculation; when finished, check to make sure that the calculation converged. You can do this by either scrolling to the bottom of the `.log` file, or by typing `$ grep 'reached' vasp.log` into the terminal.

3. The updated structural parameters appear in the CONTCAR file. It is often useful to compare the initial structure with the updated structure; type `$ diff POSCAR CONTCAR` in the terminal.

4. To have a reference of the initial starting structure, copy the POSCAR to a new file named something like POSCAR_inital: `$ cp POSCAR POSCAR_initial`

5. Copy the CONTCAR file to the POSCAR: `$ cp CONTCAR POSCAR`

6. Perform a second relaxation run using `ISTART = 1` to make sure the minimization has fully converged.

7. To accurately obtain the total energy, perform a **static** calculation – **NEVER** use energies determined from a relaxation run in which the volume has been allowed to change. Use the following settings in the INCAR:

```
PREC = Accurate
ISMEAR = -5

########## ionic relaxation parameters
##NSW = 60  # max steps for optimization ; NSW = 0 (default) for static calc
##IBRION = 1  # 1 when close to minima, 2 for difficult relaxations
##ISIF = 3 # determines type of relaxation performed (3 --> ions, shape, and vol)
##EDIFFG = -0.01  # convergence criterion: forces on all atoms < 0.01 eV/Angst
##NELMIN = 4 # minimum number of electronic steps, 4-8 for ionic relaxations
```

   Notice that the parameters related to ionic relaxation have simply been commented out: I prefer this to deleting them altogether – which would accomplish the same thing since we have now just reverted to the default settings for NSW etc. – as it helps to keep a record of your work.

8. Now you must check whether the newly optimized structure is converged with respect to the $k$-mesh sampling and energy cutoff.

   This is accomplished by performing **static** calculations (`NSW = 0`) with finer $k$-point meshes and/or increased ENCUT values and checking to make sure the forces on all of the atoms remain below the convergence criterion specified in your relaxation. If the forces increase significantly then you must relax the structure again with finer $k$ sampling and/or an increased energy cutoff.

   I highly recommend performing these checks in new directories so that you do not write over any of the files you might want to refer to from your converged relaxation. Copy the optimized POSCAR (along with the requisite KPOINTS, POTCAR, and INCAR files, of course) into a new directory (`$ cp POSCAR POTCAR KPOINTS INCAR ./newdirectory`), run a fresh calculation from scratch, and check the OUTCAR to see if forces on the atoms have increased.

## 10.2 Volume constrained structural relaxations

Here the procedure calls for relaxing the internal parameters (cell shape and ionic positions) at a series of fixed volumes (`ISIF = 4`), and then fitting the total energies as a function of volume to an equation of state so as to obtain the equilibrium volume and groundstate energy.

Although it may seem like this method might be more computationally intensive (i.e. expensive) than simply allowing the structure to relax entirely on its own, the difference in computational time typically

ends up being negligible. Additionally, there are a number of advantages to performing volume constrained relaxations: (1) This method is not succeptible to errors associated with the Pulay stress – an error occuring in the stress tensor that arises from using a constant basis set, as is the case when `ISIF = 3`. (2)

For each selected volume:

1. Make a copy of the initial `POSCAR` file named something like `POSCAR_inital` (`$ cp POSCAR POSCAR_initial`) as a reference to compare with the soon to be relaxed `CONTCAR`.

2. Relax the starting structure using `ISIF = 4`. The ionic relaxation portion of the `INCAR` should look something like this:

```
ISMEAR =  # 0 for insulators and semiconductors, 1 for metals

########## ionic relaxation parameters
NSW = 60  # max steps for optimization ; NSW = 0 (default) for static calc
IBRION = 1  # 1 when close to minima, 2 for difficult relaxations
ISIF = 4 # relaxation type (4 --> ions and cell shape optimized; vol fixed)
EDIFFG = -0.01  # convergence criterion: forces on all atoms < 0.01 eV/Angst
NELMIN = 4 # minimum number of electronic steps, 4-8 for ionic relaxations
```

3. Copy the `CONTCAR` file containing the updated structure parameters to the `POSCAR` (`$ cp CONTCAR POSCAR`) and relax the structure again (using `ISTART = 1`) to make sure it has converged.

4. To accurately obtain the total energy, perform a **static** calculation with the following settings in the `INCAR`:

```
ISMEAR = -5

########## ionic relaxation parameters
##NSW = 60  # max steps for optimization ; NSW = 0 (default) for static calc
##IBRION = 1  # 1 when close to minima, 2 for difficult relaxations
##ISIF = 3 # determines type of relaxation performed (3 --> ions, shape, and vol)
##EDIFFG = -0.01  # convergence criterion: forces on all atoms < 0.01 eV/Angst
##NELMIN = 4 # minimum number of electronic steps, 4-8 for ionic relaxations
```

   Note that the ionic relaxation parameters have been commented out, which reverts to the default settings for all of these parameters, including `NSW = 0` (no relaxation).

5. Repeat these steps for a series of cell volumes. This is best accomplished by constructing a script. Note that the following example script is quite crude and that I am at best a novice at scripting; a minimal knowledge of scripting should allow you to essentially automate all of the aforementioned steps. This script *does not inlcude a step in which an accurate total energy calculation is performed* after each relaxation `ISMEAR = -5`.

```
#! /bin/bash
BIN=/usr/local/bin/vasp
```

```
rm -f WAVECAR
for i in 0.97 0.98 0.99 1.0 1.01 1.02 1.03 ; do
cat >POSCAR <<!
O1 Pb1
$i
        3.9600000381          0.0000000000          0.0000000000
        0.0000000000          3.9600000381          0.0000000000
        0.0000000000          0.0000000000          5.0100002289
   Pb    O
    2    2
Direct
     0.250000000          0.250000000          0.237000003
     0.750000000          0.750000000          0.763000011
     0.750000000          0.250000000          0.000000000
     0.250000000          0.750000000          0.000000000
!
echo "scale= $i" ; $BIN
V=‘grep -m 1 ’volume of cell’ OUTCAR‘
E=‘tail -1 OSZICAR‘
echo "$V $E" >>SUMMARY.isif4
done
cat SUMMARY.isif4
```

Save the script as a `.sh` file, something like `EVloop.sh`

To execute the script: `$ nohup ./EVloop.sh >& vasp.log &`

In short the script does the following:

- `#! /bin/bash` : tells the script that it is a BASH script
- `BIN=/usr/local/bin/vasp` : provides the location of the VASP executable
- `rm -f WAVECAR` : removes the `WAVECAR` from the working directory. (FYI, the `-f` option is inlcuded because I have created an alias for `rm` that uses `rm -i` whenever I execute `rm`, requiring confirmation before removing files.)
- `for i in 0.97 0.98 ... ; do` : establishes `i` as a variable, and then "does" everything following the `do` for the first numerical value of the variable `i` until the script reaches `done`. The script loops between `do` and `done` for each of the variables sequentially.
- Between the `do` and `done`:

  We write over any pre-existing `POSCAR` with the `POSCAR` described between the exclamation points (!), in which the variable `i` – describing the global scaling factor for the lattice parameter matrix – is substituted.

  We "print" the scaling factor to the terminal window (`echo "scale= $i"`), and then run the VASP executable.

  Once VASP finishes running, we specify two output variables whose values will be needed: the unit cell volume, `V`, and energy, `E`. `V=‘grep -m 1 ’volume of cell’ OUTCAR‘` tells the script to search for the first occurance of `volume of cell` present in the `OUTCAR` file.

`E=‘tail -1 OSZICAR‘` tells the script to take the last line of the `OSZICAR` file. (*It is important to note that the symbol enclosing the* `grep` *and* `tail` *strings is a grave accent – located on the tilde button, whereas apostrophes enclose* `volume of cell`.)

`echo "$V $E" >>SUMMARY.isif4` now prints the values of `V` and `E` into a new file, `SUMMARY.isif4`.

The preceeding is repeated sequentially for each value of `i`

- The final line of the script, `cat SUMMARY.isif4`, prints the compiled volume and energy table to the bottom of the `vasp.log` file (which is probably unnecessary – recall here that I'm a scripting noob.)

Here is a representative example of what is contained in the `SUMMARY.isif4` document:

```
volume of cell : 105.47   17 F= -.78661919E+02 E0= -.78576836E+02  d E =-.760554E-05
volume of cell : 108.87   15 F= -.79059986E+02 E0= -.78972791E+02  d E =-.294291E-04
volume of cell : 112.35   18 F= -.79305169E+02 E0= -.79216276E+02  d E =-.549019E-05
volume of cell : 115.90   12 F= -.79411635E+02 E0= -.79321418E+02  d E =-.304376E-04
volume of cell : 119.52   14 F= -.79395432E+02 E0= -.79304298E+02  d E =-.584471E-06
volume of cell : 123.21   21 F= -.79270550E+02 E0= -.79178728E+02  d E =0.246985E-04
volume of cell : 126.99   15 F= -.79050256E+02 E0= -.78957923E+02  d E =0.619704E-05
```

Note that the total energy, the energy of interest, is the number following `F=`.

6. Now plot the total energies as a function of cell volume and fit to an equation of state to determine the equilibrim cell volume. The ELK software contains a nice utility for doing this, `eos`.

   The input file for `eos` **must contain the volumes and energies in atomic units** (Bohr$^3$, Hartree), and should be named `eos.in`. Its format is as follows:

```
"Fe3PO4O3"             : name of the compound
 11                    : number of atoms in the unit cell
 1                     : type of equation of state
 650  950       1000   : plotting range for fitted equation, number of points
 7                     : number of volume/energy pairs
711.746685899158  -2.89077074314442
734.69101824065   -2.90539942818084
758.175217225471  -2.91440973268557
782.131799523205  -2.91832228404272
806.560765133852  -2.91772683507648
831.462114057412  -2.91313750854421
856.970812954718  -2.9050418574568
```

   The output files `EVPAP.OUT` and `EVPAI.OUT` contain the datapoints and fit, respectively. Note that `eos` **normalizes the units by the number of atoms in the unit cell** – i.e. the volume is given in Bohr$^3$/atom.

7. Now check whether the equilibrium cell volume you have determined provides convergence with respect to the $k$-mesh sampling. If the forces on the atoms are above the convergence criterion you imposed during the relaxation then you must repeat the optimization using a finer $k$-mesh.

**A few other notes about relaxations:**

If the cell shape is allowed to change during a relaxation `ISIF = 3` or `ISIF = 4`, then it is *very important* to **check the optimized structure(s) to determine whether the symmetry has been retained or broken**. This is relatively painless with the aid of ACONVASP.

When performing fixed cell volume relaxations, if you do not allow the cell shape to change (i.e. if the lattice symmetry if forcefully retained) – `ISIF = 2` – **then you theoretically should not need to re-relax the structure to check for convergence** (at least according to the VASP masters: VASP ONLINE). However, bear in mind that an accurate calculation of the total energy should still be performed with `PREC = Accurate` and `ISMEAR = -5`.

If you are planning to perform phonon calculations, then relaxation should probably be converged to much stricter criteria, such as:

```
EDIFF = 1.e-6
EDIFFG = 1.e-4
```

or even

```
EDIFF = 1.e-8
EDIFFG = 1.e-6
```

which would effectively lead to convergence of the atomic positions to ca. 0.1 pm. For routine work, however, this would be complete overkill and a waste of computational resources!

**The reason this is important is because non-zero forces lead to imaginary frequencies!**

## 10.3 Relaxations using HSE06

To be completed at a later date. If you have an iminent HSE problem that requires optimizaion, I would be happy to guide you along...

## 10.4 Better constant volume optimization script example

```
#! /bin/bash
BIN=/usr/local/bin/vasp
cp POSCAR POSCAR_initial
for i in ‘seq 11.9 0.025 12.20‘; do
cat >POSCAR <<!
Mn5Ni2Bi4
$i
0.0 0.5 0.5
0.5 0.0 0.5
0.5 0.5 0.0
Mn Ni Bi
10 4 8
Cartesian
```

```
0.07500000 0.07500000 0.07500000
0.07500000 -.07500000 -.07500000
-.07500000 0.07500000 -.07500000
-.07500000 -.07500000 0.07500000
0.25000000 0.25000000 0.03600000
0.25000000 0.03600000 0.25000000
0.03600000 0.25000000 0.25000000
0.46400000 0.25000000 0.25000000
0.25000000 0.46400000 0.25000000
0.25000000 0.25000000 0.46400000
0.10300000 0.10300000 -.10300000
0.10300000 -.10300000 0.10300000
-.10300000 0.10300000 0.10300000
-.10300000 -.10300000 -.10300000
0.25000000 0.25000000 0.25000000
-.25000000 -.25000000 -.25000000
0.27200000 0.00000000 0.00000000
-.27200000 -.00000000 -.00000000
-.00000000 0.27200000 -.00000000
0.00000000 -.27200000 0.00000000
-.00000000 0.00000000 0.27200000
0.00000000 -.00000000 -.27200000
!
echo "scale= $i"
$BIN
mv CONTCAR POSCAR
echo "scale= $i"
$BIN
#V=‘grep -m 1 ’volume of cell’ OUTCAR‘
#E=‘tail -1 OSZICAR‘
#echo "$V $E" >>EVsummary.out
mv CONTCAR POSCAR
cp POSCAR POSCAR_$i
sed -i ’s/NSW/\#NSW/’ INCAR
sed -i ’s/IBRION/\#IBRION/’ INCAR
sed -i ’s/ISIF/\#ISIF/’ INCAR
sed -i ’s/EDIFFG/\#EDIFFG/’ INCAR
sed -e ’s/ISMEAR\s=\s1/ISMEAR = -5/’ INCAR
sed -i ’s/M/G/’ KPOINTS
sed -i ’s/6/8/g’ KPOINTS
echo "scale= $i"
$BIN
Vtet=‘grep -m 1 ’volume of cell’ OUTCAR‘
Etet=‘tail -1 OSZICAR‘
echo "$Vtet $Etet" >>EVsummary.out
```

```
sed -i 's/\#NSW/NSW/' INCAR
sed -i 's/\#IBRION/IBRION/' INCAR
sed -i 's/\#ISIF/ISIF/' INCAR
sed -i 's/\#EDIFFG/EDIFFG/' INCAR
sed -e 's/ISMEAR\s=\s-5/ISMEAR = 1/' INCAR
sed -i 's/G/M/' KPOINTS
sed -i 's/8/6/g' KPOINTS
done
```

# 11 VASP crib sheet