

AFLOW V 3.1.155

```
*****
*
*          aflow - STEFANO CURTAROLO Duke University 2003-2018
*          High-Throughput ab-initio Computing Project
*
*****
LATEST VERSION OF THE FILE:          materials.duke.edu/AFLOW/aflow_aconvasp.pdf
*****
```

AFLOW POSTPROCESSING MODE (formerly called aconvasp/aqe)

aflow --help [-h]

Gives Help information.

aflow --version | -v

Gives Version Information.

aflow --machine

Gives Machine Information.

aflow --check_integrity | --checki

Check the integrity of xlib functions

aflow --abccar < POSCAR | WYCCAR

Converts the POSCAR or WYCCAR in format.

POSCAR is the usual VASP file

WYCCAR is described as

TITLE

SCALE (positive (rescaling) negative (volume))

A B C ALPHA BETA GAMMA SG# [OPTION#]

#specie0 #specie1

DIRECT (or CARTESIAN)

.. specie0

.. specie0

..

.. specie1

.. specie1

and so on. (Stefano Feb 2009)

The positions of the species will be used with the list of

symmetry operations (aflow_wyckoff.cpp) to generate all the atoms.

aflow --abinit

Transforms the POSCAR (or whichever is the input file format) to a
ABINIT GEOM format.

The geometrical file follows the ABINIT convention.

aflow --aims

Transforms the POSCAR (or whichever is the input file format) to a
AIMS GEOM format.

The geometrical file follows the AIMS convention.

aflow --ace < POSCAR

Outputs to standard out a cell standard ASCII (ace) file
based on the POSCAR input file. This can be used as input
for CarIne.

aflow --use_aflow.in=XXX

Uses XXX instead of "aflow.in" in searching/running/operating directories.

The option is very useful for compounded calculations.

aflow --aflowin < POSCAR

Output the structure inside the strings:

[VASP_POSCAR_MODE_EXPLICIT]START

structure...

[VASP_POSCAR_MODE_EXPLICIT]STOP

which is useful if you want to recycle partially ran aflow.in !

```

aflow [options] --aflow_proto=label*:speciesA*[:speciesB*][:volumeA*[:volumeB*]|:volume] [--params=.... [--h
Creates automatic aflow.in.
OPTIONS are:
--usage
--potential=pot_LDA | pot_GGA | potpaw_LDA | potpaw_GGA | potpaw_PBE
--potential_complete
--missing
--noautopp
--kppra=NNNN (default: DEFAULT_KPPRA in .aflow.rc) --kppra_static=NNNN (default: DEFAULT_KPPRA_STATIC in .aflo
--enmax_multiply=NNNN (default: DEFAULT_VASP_PREC_ENMAX_HIGH in .aflow.rc, DEFAULT_VASP_PREC_ENMAX_ACC
--pressure=0,1,2 (string of kB pressures separated by commas)
--potim=XXX (default: DEFAULT_VASP_PREC_POTIM in .aflow.rc) (VASP)
--relax_mode=[ENERGY | FORCES | ENERGY_FORCES | FORCES_ENERGY] (default: DEFAULT_VASP_FORCE_OPTION_REL
--precision=[LOW | MEDIUM | NORMAL | HIGH | ACCURATE], PRESERVED] (default: DEFAULT_VASP_FORCE_OPTION
--metagga=[TPSS | RTPSS | MO6L | MBJL | SCAN | MS0 | MS1 | MS2 | NONE] (default: DEFAULT_VASP_FORCE_OP
--ivdw=[number_for_VASP_see_manual_for_IVDW | 0] (default: DEFAULT_VASP_FORCE_OPTION_IVDW_SCHEME in .aflow.rc)
--algorithm=[(NORMAL | VERYFAST | FAST | ALL | DAMPED), PRESERVED] (default: DEFAULT_VASP_FORCE_OPTION_ALGO_SC
--type=[METAL | INSULATOR | SEMICONDUCTOR | DEFAULT] (default: DEFAULT_VASP_FORCE_OPTION_TYPE_SCHEME i
--convert_unit_cell= (SPRIM, SCONV, NIGGLI, MINK, INCELL, COMPACT, WS, CART, FRAC, PRES)
--volume_plus_equal=XXX
--volume_multiply_equal=XXX
--ediffg=XXX (default: DEFAULT_VASP_PREC_EDIFFG_XXXX in .aflow.rc) (VASP)
--ldau2
--noldau2
--bands
--neglect_nomix
--stdout
--qe
--abinit
--aims
--list
--params=.... { check aflow --readme=anrl }
--hex          { check aflow --readme=anrl }

```

This command generates the directory: `./AFLOWDATA/speciesAspeciesB/label/` and create an `aflow.in` inside with the "label" structure from HTQC library project, with `speciesAs`, `speciesBs`,.. and the volumes per atoms, `volumeAs`, `volumeBs`,.. (real in Å^3).

Aflow generates a huge number, `#speciesAs*#speciesBs...#label[*#pressures]`, of directories containing all the combinations of `./AFLOWDATA/speciesAspeciesB./label:pressure` and so on. This is helpful for generating huge databases.

Be careful to include the various "`_pv`," "`_sv`," etc" in the species so that the POTCAR makes sense.

HIGH-THROUGHPUT NOTE: `label`, `speciesA`, `speciesB`, `volumeA`, `volumeB` can be multiple strings separated by commas without spaces in between such as

Note that that `--aflow_proto label A B ...` is not supported anymore and you now need to use `--aflow_proto=label:A:B... et cetera`.

LABEL

`label*` = string of `label1`,`label2`,`label3`,... separated by commas ","
specify which labels/prototypes to generate

SPECIES

`speciesA*` = string of `speciesA1`,`speciesA2`,`speciesA3`,... separated by commas ","
specify which species in position A to consider.

`speciesB*` = string of `speciesB1`,`speciesB2`,`speciesB3`,... separated by commas ","
specify which species in position B to consider.

similarly you can have `speciesC*`,`speciesD*`, and so on.

VOLUMES

`volumeA*` = string of `volumeA1`,`volumeA2`,`volumeA3`,... separated by commas ","
specify the volumes of `speciesAs`, the number must be identical to the number of `speciesAs` otherwise you get an error.

volumeB* = string of volumeB1,volumeB2,volumeB3,... separated by commas ","
specify the volumes of speciesBs, the number must be identical to the number
of speciesBs otherwise you get an error.
similarly you can have volumeC*,volumeD*, and so on.
If you want to set one volume per atom, identical to all the species, then
specify only one number as species...:volume .

USAGE

With --usage a brief syntax help comes out.

POTENTIALS

The user can specify the potential type.
--potential=pot_LDA | pot_GGA | potpaw_LDA | potpaw_GGA | potpaw_PBE
If not specified it takes potpaw_PBE and the standard default for the species.
With --potential_complete it creates the subdirectory containing the type and
date of the potentials: i.e. Ti_sv => Ti_sv:PAW_GGA:07Sep2000
With the option --noautopp, then you have to be careful to include the various
"_pv,_sv, etc" in the species so that the POTCAR makes sense.

MISSING

With the option --missing it creates automatic aflow.in only for the missing .
structures of the database. It checks on:

\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB1))/LIB/	of AFLOW_MATERIALS_SERVER
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB2))/LIB/	of AFLOW_MATERIALS_SERVER
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB3))/LIB/	of AFLOW_MATERIALS_SERVER
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB4))/LIB/	of AFLOW_MATERIALS_SERVER
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB5))/LIB/	of AFLOW_MATERIALS_SERVER
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB6))/LIB/	of AFLOW_MATERIALS_SERVER
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICDS))/LIB/	of AFLOW_MATERIALS_SERVER
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_AURO))/LIB/	of AFLOW_MATERIALS_SERVER

KPOINTS SPECIFICATION

With --kppra=NNN the user can specify a NNN number of kpoints per reciprocal atom.
Input KPPRA (6,000-10,000 recommended) is the lower bound of the product of the
(total count of k-points in the 3-D grid) and
(number of atoms in the cell).
The smallest mesh to satisfy this requirement is calculated.
Effectively, cells with more atoms will yield smaller grid meshes.

PRESSURE CALCULATIONS

The user can run calculations under pressure by adding --pressure (kB).

--pressure=p0[,p1][,p1]...

Example

--pressure=10,20,30

Then the data will be generated as directories containing all the combinations of

./AFLOWDATA/specieAspecieB/label:P=p0/

./AFLOWDATA/specieAspecieB/label:P=p1/

The input file aflow.in will contain the correct PSTRESS=** entry

BANDS CALCULATIONS

The user can run bands by adding --bands

POTIM PARAMETER

For some systems VASP requires POTIM smaller than the default

0.5. See the VASP Manual.

RELAX_MODE

One of the RELAX_MODEs specified in the AFLOW manual (VASP only).

[ENERGY | FORCES | ENERGY_FORCES | FORCES_ENERGY] (default: DEFAULT_VASP_FORCE_OPTION_RELAX_MODE_SCHEME in .af1)

PRECISION

One of the PRECISIONs specified in the AFLOW manual (VASP only).

[(LOW | MEDIUM | NORMAL | HIGH | ACCURATE), PRESERVED] (default: DEFAULT_VASP_FORCE_OPTION_PREC_SCHEME)

ALGORITHM

One of the ALGORITHMs specified in the AFLOW manual (VASP only).
 [(NORMAL | VERYFAST | FAST | ALL | DAMPED), PRESERVED] (default: DEFAULT_VASP_FORCE_OPTION_ALGO_SCHEME)

TYPE
 One of the TYPEs specified in the AFLOW manual (VASP only).
 [METAL | INSULATOR | SEMICONDUCTOR | DEFAULT] (default: DEFAULT_VASP_FORCE_OPTION_TYPE_SCHEME in .aflow)

CONVERT_UNIT_CELL
 One of the CONVERSIONs specified in the AFLOW manual (VASP only).
 [SPRIM, SCONV, NIGGLI, MINK, INCELL, COMPACT, WS, CART, FRAC, PRES]

VOLUME CHANGES
 Volume plus equal adds XXX to the current volume.
 Volume multiply equal multiplies XXX to the current volume.

POTIM
 --potim=XXX parameter overrides the default value (default: DEFAULT_VASP_PREC_POTIM in .aflow.rc) for a

EDIFFG
 --ediffg=XXX parameter overrides the default value (default: DEFAULT_VASP_PREC_EDIFFG_XXXX in .aflow.rc)

LDAU CALCULATIONS (FORCING IT ON AND OFF)
 The user can run bands by adding --ldau2 and the code will take the default ldau parameters that are specified inside aflow_avasp.cpp.
 If you do not like them, you can modify the cpp code OR change the lines inside aflow.in
 [VASP_FORCE_OPTION]LDAU2=ON
 [VASP_FORCE_OPTION]LDAU_PARAMETERS=Bi,Li,Zn;1,-1,2;0,0,7.5;0,0,0 // species;Ls;Us;Js
 For some reference systems, LDAU is taken by default. Then the user can remove it with --noldau.

MIXING (TO MIX OR NOT TO MIX)
 Some systems is known to be no-mix, and they will be skipped from the calculations.
 To override this automatism, specify --neglect_nomix .

STDOUT output
 If you specify --stdout, aflow will not write the file but print on the screen.

QUANTUM ESPRESSO output
 If you specify --qe, aflow will add QUANTUM ESPRESSO code to aflow.in (experimental).

ABINIT output
 If you specify --abinit, aflow will add ABINIT code to aflow.in (experimental).

AIMS output
 If you specify --aims, aflow will add AIMS code to aflow.in (experimental).

LIST output
 If you specify --list, aflow will not generate the aflow.ins but only report the ones to be created (useful for scripting).

ANRL: AFLOW NAVAL RESEARCH LAB PROTOTYPE LIBRARY
 With respect to the article:
 Mehl et al., "The AFLOW Library of Crystallographic Prototype", <https://arxiv.org/abs/1607.02532>
 the code gives the possibility to define the parameters of the cell --params=...
 Some rhombohedral cells can be generated in hexagonal configurations (see the article).
 The choice is performed with the flag --hex.
 Check aflow --readme=anrl

aflow --aflow_proto=label[:]specieA[:]specieB[:]volume
 Same as above but the volume per atoms are extracted from the most dense paw_pbe values (usually the ground state) and with the Vegard's law (Vegard's law is an approximate empirical rule which holds that a linear relation exists, at constant temperature, between the crystal lattice constant of an alloy and the concentrations of the constituent elements).

HIGH-THROUGHPUT NOTE: label, specieA, specieB can be multiple strings separated by commas without spaces in between such as label1,label2,label3 generate all the labels
 specieA1,specieA2,specieA3 goes through all specieAs times all Bs
 specieB1,specieB2 goes through all specieBs times all As
 In this case aflow generates a huge number, #labels*#specieAs*#specieB of directories containing all the combinations of ./AFLOWDATA/specieA?specieB?/label? and so on. This is helpful for generating huge databases. PRESSURE, POTENTIALS, BANDS calculations are allowed, see above. STDOUT option available.

aflow --aflowSG[_label,_number][=tolerance | =tight | =loose] [--no_scan] [--mag|--magnetic|--magmom=[m1,m2,. . .]]

Calculates the space group of the crystal in the ITC convention. No assumptions are made on the input POSCAR-- may be supercell, primitive, etc. Gives the Bravais lattice, crystal system, point group, space group number and wyckoff set. The origin choice is always the first choice in ITC 5th edition. This is the highest symmetry choice. (However, for Monoclinic systems, different unique axes and cell choices are examined to ensure a spacegroup is identified.) This routine consists of an adaptive tolerance; if any symmetry rules are broken, the tolerance is changed and everything is recalculated.

Tolerance:

The tolerance is given in Angstroms. There are two preset tolerances:
 tight: minimum_interatomic_distance/100.0
 loose: minimum_interatomic_distance/10.0

The default tolerance is the "tight" tolerance value.

A value can also be specified. (Note, it must be below the minimum interatomic distance, otherwise an error will be thrown.)

Can be called inside aflow via:

```
xstructure.SpaceGroup_ITC()           (default tolerance)
xstructure.SpaceGroup_ITC(tol)       (user defined tolerance)
```

AFLOW counterpart to platonSG and findsymSG.

Magnetic analysis:

The magnetic tag specifies the magnetic moment on each of the atoms. The two input formats are a comma separated string of the magnetic moments - in the same order as the input file, the location of the INCAR, or the location of the OUTCAR, which is then read and the magnetic moment for each atom is found (must be a static calculation, otherwise this information isn't specified).

Works for both collinear and non-collinear systems (AFLOW autodetects collinear/non-collinear based on the input).

Options:

[--no_scan] : Will not perform tolerance scan.

[--mag|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]]: Perform symmetry analysis of spin system, where

aflow --alphabetic < POSCAR

Makes the structure in alphabetic order (if possible).

aflow --alpha_compound=string1,string2,...

Makes the alphabetic "compound" from the name of string. For instance

```
aflow --alpha_compound=Zr15.6Ag2Mg14
returns Ag2Mg14Zr15.6
```

aflow --alpha_species=string1,string2,...

Makes the alphabetic "species" from the name of string. For instance

```
aflow --alpha_species=Zr15.6Ag2Mg14
returns AgMgZr
```

aflow --angle=cutoff < POSCAR

Outputs to standard out the angles for each atom triplet made up of neighbors within a distance cutoff of each other.

Actually, since this can be huge, only the first MAX_NUM_ANGLE-1 neighbours within d are used. MAX_NUM_ANGLE=21 at present and can be set in aflow_pflow_print.cpp

aflow --bader -D DIRECTORY

options:

```

[ --usage ]
[ --critical_points|--cp ]
[ --calculate=|--calc=bader|voronoi ]
[ --nocalculate=|--nocalc=bader|voronoi ]
[ --partition=|--part=neargrid|ongrid ]
[ --refine_edge_method=|--rem=-1|-2|-3 ]
[ --reference=|--ref=REF_FILE ]
[ --vacuum=|--vac=off|auto|DENSITY_THRESHOLD ]
[ --terminate=|--term=known|max ]
[ --print_all=atom|bader|both ]
[ --print_index=|--print_idx=atom|bader|both ]
[ --print_select_atom=|--print_sel_atom=[LIST OR RANGE] ]
[ --print_select_bader=|--print_sel_bader=[LIST OR RANGE] ]
[ --print_sum_atom=[LIST OR RANGE] ]
[ --print_sum_bader=[LIST OR RANGE] ]
[ --quiet|--q ]
[ --consolidate_atoms2species|--a2s ]
[ --remove_bader_atoms|--rba ]
[ --jvxl_all_species=|--jvxl=CUTOFF1,CUTOFF2...[:DOWNSAMPLE1,DOWNSAMPLE2,..]
                                |CUTOFF1[,DOWNSAMPLE1:CUTOFF2,DOWNSAMPLE2:...]
                                |CUTOFF[,DOWNSAMPLE] ]
[ --keep=jvxl_only|--jvxl_only ]
[ -D DIRECTORY ]

```

Perform Bader charge analysis by means of FORTRAN code from the Henkelman Group at UT, Austin. <http://theory.cm.utexas.edu/bader/>

DIRECTORY is the path to the data from VASP.

If this is omitted, it will assume the desired path is the one from which the command is called.

The following files are required to be present in DIRECTORY:

CHGCAR.static|CHGCAR

OUTCAR.static|OUTCAR

POSCAR.static|POSCAR

[AECCARO.static|AECCARO] } OR A CORRESPONDING REFERENCE CHARGE DENSITY FILE

[AECCAR2.static|AECCAR2] } (explained below)

though they can be compressed as .bz2 or .gz files.

Description of flags and, if applicable, corresponding bader code command:

--calculate=|--calc=bader|voronoi

--nocalculate=|--nocalc=bader|voronoi

[corresponding bader code command: -c | -n < bader | voronoi >]

Turn on [-c] or off [-n] the following calculations

bader: Bader atoms in molecules (default)

voronoi: population analysis based on distance

WARNING: If you specify --nocalc=bader without a --calc command, the bader code is not expected to produce any files.

It is sufficient to specify --calc=voronoi without --nocalc=bader.

EXPECTED FILES: ACF.dat, AVF.dat, BCF.dat, LABEL_abader.out (net charges and atomic volumes calculated from bader code).

--partition=|--part=neargrid|ongrid

[corresponding bader code command -b < neargrid | ongrid >]

Use the default near-grid bader partitioning or the original on-grid based algorithm.

--refine_edge_method=|--rem=-1|-2|-3

[corresponding bader code command -r < refine_edge_method >]

By default (-r -1), only the points around reassigned

points are checked during refinements. The old method,

which checks every edge point during each refinement, can

be enabled using the -r -2 switch:

A new weight method developed by Yu and Trinkle and be

enabled with the -r -3 switch.

```

--reference=|--ref=REF_FILE
[ corresponding bader code command -ref REF_FILE ]
    By default, AFLOW will sum the AECCARO and AECCAR2 to form the aflow.CHGCAR_sum file.
    However, this can be overridden by specifying a desired reference file.
EXPECTED FILES: aflow.CHGCAR_sum

--vacuum=|--vac=off|auto|DENSITY_THRESHOLD
[ corresponding bader code command -vac < off | auto | vacuum_density > ]
    Assign low density points to vacuum.
    auto: vacuum density cutoff is 1E-3 e/Ang^3 (by default in AFLOW)
    off: do not assign low density points to a vacuum volume
    DENSITY_THRESHOLD: a float designating the maximum density assigned to a vacuum volume

--terminate=|--term=known|max
[ corresponding bader code command -m < known | max > ]
    Determine how trajectories terminate
    known: stop when a point is surrounded by known points
    max: stop only when a charge density maximum is reached

--print_all=atom|bader|both
[ corresponding bader code command -p < all_atom | all_bader > ]
    Print calculated volumes (containing charge above threshold of 0.0001 electrons)
    atom: all atomic volumes
    bader: all Bader volumes
EXPECTED FILES: BvAtxxxx.dat/Bvolxxxx.dat

--print_index=|--print_idx=atom|bader|both
[ corresponding bader code command -p < atom_index | bader_index > ]
    Print corresponding volume index
    atom: atomic volume index
    bader: bader volume index
EXPECTED FILES: AtIndex.dat/BvIndex.dat

--print_select_atom=|--print_sel_atom=[LIST OR RANGE]
[ corresponding bader code command -p sel_atom [LIST OR RANGE] ]
    Print specified atomic volumes.
    List or range corresponds to 1,2,3 (comma-separated) or 1-3 (hyphen-separated).
EXPECTED FILES: BvAtxxxx.dat

--print_select_bader=|--print_sel_bader=[LIST OR RANGE]
[ corresponding bader code command -p sel_bader [LIST OR RANGE] ]
    Print specified bader volumes.
    List or range corresponds to 1,2,3 (comma-separated) or 1-3 (hyphen-separated).
EXPECTED FILES: B_wexxxx.dat (artifact of bader code)

--print_sum_atom=[LIST OR RANGE]
[ corresponding bader code command -p sum_atom [LIST OR RANGE] ]
    Print sum of specified atomic volumes.
    Consider using aflow --chgsum command instead.
    List or range corresponds to 1,2,3 (comma-separated) or 1-3 (hyphen-separated).
EXPECTED FILES: BvAt_summed.at

--print_sum_bader=[LIST OR RANGE]
[ corresponding bader code command -p sum_bader [LIST OR RANGE] ]
    Print sum of specified bader volumes.
    Consider using aflow --chgsum command instead.
    List or range corresponds to 1,2,3 (comma-separated) or 1-3 (hyphen-separated).
EXPECTED FILES: Bvol_summed.at

--quiet|--q
    No output printed to screen.

--consolidate_atoms2species|--a2s
    Combine all atomic volumes to form species volumes.

```

EXPECTED FILES: BvAt_SPECIES.dat

--remove_bader_atoms|--rba

Remove individual atomic volumes.
Useful if you only want to keep species volumes.
Only read if --a2s specified.

--jvxl_all_species=|--jvxl=CUTOFF1,CUTOFF2...[:DOWNSAMPLE1,DOWNSAMPLE2,...]
|CUTOFF1[,DOWNSAMPLE1:CUTOFF2,DOWNSAMPLE2:...]
|CUTOFF[,DOWNSAMPLE]

Generate .jvxl files of all species volumes at the specified cutoffs.
A .jvxl file is a space-efficient file format for surface charges read by Jmol.
See <http://chemapps.stolaf.edu/jmol/docs/misc/JVXL-format.pdf>.
You may also specify a downsample ratio.

This will reduce the number of points on the surface by the specified factor.
You may produce multiple .jvxl files in the following ways:

CYCLIC MODE (:)

Every combination of cutoff [and downsample_ratio] will be produced

SETS MODE (:)

Provide sets of cutoff [and corresponding downsample_ratio] to be produced

WARNING: while you may specify a cutoff of arbitrary precision, the label of the
output file will only show precision to the nearest hundredth.

EXPECTED FILES: LABEL_Bader_CUTOFF_DOWNSAMPLE_RATIO_SPECIES.jvxl

--keep=jvxl_only|--jvxl_only

After producing .jvxl files, remove atomic and species volume files.
Only read if --jvxl specified.

NOTE:

Multiple print command can be issued, with the exception of:

--print_all=atom && --print_sel_atom=[LIST OR RANGE]

--print_all=bader && --print_sel_bader=[LIST OR RANGE]

By default, the program will ignore the latter command and only
execute the former (--print_all).

All CHG files printed from bader code will have its header amended
so that it may be read by Jmol.

See --prepare_chgcar_4_jmol command for more information.

Bader Charge code reference:

<http://theory.cm.utexas.edu/henkelman/code/bader/>

W. Tang, E. Sanville, and G. Henkelman. J. Phys.: Condens. Matter 21, 084204 (2009).

E. Sanville, S. D. Kenny, R. Smith, and G. Henkelman. J. Comp. Chem. 28, 899-908 (2007).

G. Henkelman, A. Arnaldsson, and H. Jonsson. Comput. Mater. Sci. 36, 254-360 (2006).

aflow --bandgap=[bands_directory1[,bands_directory2,...]]

Calculates the energy gap and band edges of a material by looking at the occupancies in the OUTCAR file

On output it produces the following background system information:

System : The system name, identical to the SYSTEM tag in the OUTCAR.

Spin tag : Whether it is a spin polarized / unpolarized calculation.

Fermi level : Obtained from the E-fermi tag in the OUTCAR.

The energy gap (Egap) and band edges (Valence Band Top, VBT, and Conduction Band Bottom, CBB) are then
output in table format, with 5 significant figures.

Unpolarized example:

	VBT	CBB	Egap	Type
Net Result :	0.0000e-00	0.0000e+00	+0.0000e+00	insulator_direct

Polarized example:

	VBT	CBB	Egap	Type
Majority Spin :	0.0000e+00	0.0000e+00	0.0000e+00	insulator_indirect
Minority Spin :	-1.0000e+00	-1.0000e+00	-1.0000e+09	metal
Net Result :	0.0000e+00	0.0000e+00	0.0000e+00	insulator_direct

Special for metallic materials: the edges are arbitrarily set to -1 and the gap is arbitrarily set to -1.0E-09. The "Net Result" is the overall gap and edges of the system. Note that the band edges have been referenced to the Fermi energy.

There are several meaningful tags (separated by "_") provided under "Type", including:
(half-)metal: no band gap present in at least one of the spin-channels
insulator: band gap present
(in)/direct: issued for insulators, specifies whether VBT and CBB come from the same k-point
zero-gap: issued for insulators with a negligible gap (tol===1e-4 eV)
spin-polarized: issued for spin-polarized insulators where VBT and CBB come from different spin-channels
empty(-partial): issued for materials where no band edge is detected for at least a single k-point (

Sample output for a Spin-polarized material, Cr102_ICSD_186838:

```
System      : Cr102_ICSD_186838
Spin tag    : 2
Fermi level : +5.2091e+00
            VBT      CBB      Egap      Egap_fit   Type
Majority Spin : -1.0000e+00 -1.0000e+00 -1.0000e+09 -1.0000e+09 metal
Minority Spin : -4.9230e-01 +2.1851e+00 +2.6774e+00 +4.5221e+00 insulator-indirect
Net Result    : -1.0000e+00 -1.0000e+00 -1.0000e+09 -1.0000e+09 half-metal
```

`aflow --bandgaps < dirlist`

Similar to `--bandgap`, the directories of bands runs are stored in file "dirlist", one directory per line (see also `--calculated=bandgaps`)

`aflow --bands=PROOUT < POSCAR`

Outputs the up and down bands to band.up.out and band.dn.out.

Format is

k-space path length (true, fractional), bands (up or down),
nkpt, kx, ky, kz

Uses data in PROOUT file. Uses POSCAR to get lattice to calculate true reciprocal distances. Must run vasp with LORBIT=2 to get the PROOUT file.

`aflow --bs | --band_structures`

Given a calculation with all the parts (static,bands) creates the subdirectory BANDS with the band structures files.

`aflow --bzdirections | --bzd < POSCAR`

Calculates lattice and print a nice looking KPOINTS in VASP format.

`aflow --bzdirections= | --bzd=LATTICE`

Print a nice looking KPOINTS in VASP format for one of the lattices, where the lattices are in their conventional-primitive form (kpoints are fractional of the reciprocal lattice of the CONVENTIONAL primitive).

LATTICES = (the ones with "")

1. TRI order: $k_{\alpha}, k_{\beta}, k_{\gamma} > 90$ ($k_{\gamma} < k_{\alpha}, k_{\gamma} < k_{\beta}$)
or $k_{\alpha}, k_{\beta}, k_{\gamma} < 90$ ($k_{\gamma} > k_{\alpha}, k_{\gamma} > k_{\beta}$)
special case when $k_{\gamma} = 90$

"TRI1a" $k_{\alpha} > 90, k_{\beta} > 90, k_{\gamma} > 90$

"TRI1b" $k_{\alpha} < 90, k_{\beta} < 90, k_{\gamma} < 90$

"TRI2a" $k_{\alpha} > 90, k_{\beta} > 90, k_{\gamma} = 90$

"TRI2b" $k_{\alpha} < 90, k_{\beta} < 90, k_{\gamma} = 90$

2. "MCL" unique (order $b \leq c$)

3. MCLC (order $\alpha < 90$)

"MCLC1" $k_{\gamma} > 90$

"MCLC2" $k_{\gamma} = 90$

"MCLC3" $k_{\gamma} < 90, b \cdot \cos(\alpha)/c + (b \cdot \sin(\alpha)/a)^2 < 1$

"MCLC4" $k_{\gamma} < 90, b \cdot \cos(\alpha)/c + (b \cdot \sin(\alpha)/a)^2 = 1$

"MCLC5" $k_{\gamma} < 90, b \cdot \cos(\alpha)/c + (b \cdot \sin(\alpha)/a)^2 > 1$

4. "ORC" unique (order $a < b < c$)

5. "ORCC" unique (order $a < b$)

6. ORCF (order $a < b < c$)

```

    "ORCF1" "ORCF_invb2+invc2<inva2" for  $1/a^2 > 1/b^2 + 1/c^2$ 
    "ORCF2" "ORCF_inva2<invb2+invc2" for  $1/a^2 < 1/b^2 + 1/c^2$ 
    "ORCF3" for  $1/a^2 = 1/b^2 + 1/c^2$ 
7. "ORCI" unique (order a<b<c)
8. "TET" unique (order a a c)
9. BCT (order a a c)
   "BCT1" "BCT_c<a" for c<a
   "BCT2" "BCT_c>a" for c>a
10. "RHL1" alpha<90
     "RHL2" alpha>90
11. "HEX" unique (order 60 90 90)
12. "CUB" unique
13. "FCC" unique (order 60 60 60)
14. "BCC" unique

aflow --BZmax < POSCAR
Calculates the distance between high-symmetry K points and Gamma point
in the reciprocal space, and sorts them.

aflow --bzplot | --plotbz < POSCAR
Reads POSCAR from stdin and makes brillouin zone image file "bzplot.eps" and "bzplot.png".
The data for plotting is saved in bzplot.dat and the script to make the plot is saved in plotbz.sh.
plotbz.sh can be modified by hand to make further adjustment of the figure.
aflow --bzplotdata < POSCAR > bzplot.dat
Reads POSCAR and generates data for brillouin zone and kpath plotting.
The data is outputted to stdout (cout).

aflow --bzplotuseKPOINTS=KPOINTS < POSCAR
aflow --bzplotdatauseKPOINTS=KPOINTS < POSCAR > bzplot.dat
Analogous to --bzplot and --bzplotdata respectively, however the first word of the first line
in KPOINTS file is used as the str_sp.bravais_lattice_variation_type to get the kpath (as opposed to
doing --sp of the POSCAR).

aflow [--np=NP] --cages[=roughness] < POSCAR
Returns the center, radius and coordination (with atoms number)
of the cages for interstitial defects (SC Nov07).
It prints all the cages generated by putting spheres between
four, three and two points, stable and metastable respectively.
After finding all the cages (REDUCIBLE) the algorithm removes the
ones symmetrically equivalent (by using the space group calculated
at the beginning). The cages that are worth simulating are the
IRREDUCIBLE ones.
Radius is the radius includeing the scale.
P is the number of points used to find the sphere.
C is the coordination of the sphere (the number of atoms on its
surface, from radius to radius+roughness.
T is the label of the irreducible cage. You can see in the last
part of the output the lables of the reduced cages.
NOTES: If you specify "--np=NP", a concurrent number of NP posix
threads will be started to speed the calculation in a multicpu-
multicore environent.

aflow --calculated=icsd --random
Print one of the available calculated directories (usefull for webs).
aflow --calculated[=[=]all|icsd|lib1|lib2|lib3|lib4|lib5|lib6|auro]
"all": print the total number of calculations in the databases.
"icsd": print the list of calculated ICSD-structures.
"lib1": print the list of calculated pure systems.
"lib2": print the list of calculated binary systems.
"lib3": print the list of calculated ternary systems (magnetic, Heusler projects, etc.)
"lib4": print the list of calculated quaternary systems (very few)
"lib5": print the list of calculated quaternary systems (very few)
"lib6": print the list of calculated quaternary systems (very few)
"auro": print the list of calculated auro database system.

```

```

aflow --cart [-c] < POSCAR
  Outputs to standard out a new POSCAR file with atom
  positions in cartesian coordinates.
aflow --chgcar2jvxl|--c2j=CHGCAR1[,CHGCAR2,...]::CUTOFF1,CUTOFF2...[::DOWNSAMPLE1,DOWNSAMPLE2,...]
  |CHGCAR1,CUTOFF1[,DOWNSAMPLE1:CHGCAR2,CUTOFF2[,DOWNSAMPLE2:...]]
  |CHGCAR,CUTOFF[,DOWNSAMPLE]

options:
  [ --usage ]
  [ --output=|--o= ]

Converts a surface data file (e.g. CHGCAR file) to a .jvxl file at a specified cutoff.
See http://chemapps.stolaf.edu/jmol/docs/misc/JVXL-format.pdf.
You may also specify a downsample ratio.
This will reduce the number of points on the surface by the specified factor.
You may produce multiple .jvxl files in the following ways:
  CYCLIC MODE (::)
    Every combination of cutoff [and downsample_ratio] will be produced
  SETS MODE (:)
    Provide sets of cutoff [and corresponding downsample_ratio] to be produced
You may specify the name of the output file (ending with .jvxl) ONLY if sets are provided.
WARNING: while you may specify a cutoff of arbitrary precision, the label of the
  output file will only show precision to the nearest hundredth.
aflow --chgdiff=CHGCAR1,CHGCAR2
options:
  [ --usage ]
  [ --output=|--o=CHGCAR_OUT ]

This takes the difference between two CHGCAR files, outputting
CHGCAR1-CHGCAR2. The output is in the form of a vasp46s
CHGCAR. The output file is aflow_CHGDIFF.out (unless otherwise
specified).

aflow --chgsum=CHGCAR1,CHGCAR2,...
options:
  [ --usage ]
  [ --output=|--o=CHGCAR_OUT ]

This finds the sum between two or more CHGCAR files. The output
is in the form of a vasp46s CHGCAR.
The output file is aflow_CHGSUM.out (unless otherwise
specified).
aflow --chgint CHGCAR
Outputs to standard out the integrated charge density around
every atom. The total integrated density within the Voronoi
volume is given for each atom. The code also calculates
the integrated density in a surrounding sphere for each atom.
This is output as a function of radius for radii from 0 to 3
angstrom in steps of 0.1 angstrom. The total,Up-Dn,Up,Dn
charges are all integrated although for non-spin polarized
calculations only the first gives new information. The code
works on all the recent versions of vasp we have tried but they
seem to change the CHGCAR formatting sometimes so it may crash
on different versions (at least 4.4.1 and 4.4.5 (including PAW)
work). For a large cell this can take a while (e.g., 80 atoms
might take ~20 minutes). vasp4631 ok.
aflow --cif < POSCAR
Outputs to standard out a Crystallographic Information File
(cif) format file based on the POSCAR input file.
This can be used as input for Mercury, and is the preferred
format in Gerd's structure collection.
aflow --clat=a,b,c,alpha,beta,gamma
Outputs to standard out the cartesian lattice vectors obtained
from the input a b c alpha beta gamma.

aflow --clean --DIRECTORY=DIRECTORY

```

```

Cleans the directory, like aflow --clean --DIRECTORY=DIRECTORY

aflow --clean_all < LIST_DIRECTORIES
Cleans all the directories. Be careful !
(if the LIST_DIRECTORIES contain aflow.in and LOCK the substring
gets removed to help the search).

aflow --cluster-expansion=... | --ce=structure_type,A,B,EA,EB
--ce structure_type A B formation_energy_A formation_energy_B
Cluster expansion
structure_type: fcc/bcc/hcp
A, B: element names
EA, EB: formation energy of pure element A and B

aflow --cluster=structure_type,atom_num_min,atom_num_max,neighbour_min,neighbour_max
Get clusters
atom_num_min, atom_num_max : minimum and maximum numbers of atoms in a cluster
neighbour_min, neighbour_max : minimum and maximum nearest neighbour
pairs including in a cluster

aflow --cmp_str POSCAR1 POSCAR2 rcut
This compares the characteristics of two POSCAR files and is
useful for determining if the files are the same. Characteristics
compared include number of atoms, number of types, total volume,
volume per atom, lattice parameters, number of neighbours of each
pair type out to rcut, differences in bond lengths for neighbours
of each pair type out to cutoff, and space groups. For rcut<0 the
cutoff is set to 4*(Wigner-Seitz radius of each atom) (this is the
radius such that num atom spheres occupies the whole volume).

aflow --compare=a,b,c,d,e,f,g,h,k,j,i,l
Outputs to standard output the % comparison between
a#g b#h c#k d#j e#i f#l in %
cout << abs(a1-a7)/((a1+a7)/2.0) << ;
cout << abs(a2-a8)/((a2+a8)/2.0) << ;
cout << abs(a3-a9)/((a3+a9)/2.0) << ;
cout << abs(a4-a10)/((a4+a10)/2.0) << ;
cout << abs(a5-a11)/((a5+a11)/2.0) << ;
cout << abs(a6-a12)/((a6+a12)/2.0) << ;
cout << endl;
Useful with y-ndata to compare % relaxations.

aflow --compare_material=POSCAR_1,POSCAR_2 [--np=xx] [--print]
Compares POSCAR_1 to POSCAR_2 if the ratios are commensurate
and types of atoms are the same (i.e. same material).
POSCARs must be in the same directory you are running in.
[--np=xx] : Number of processors (default 8). Algorithm is thread-friendly.
[--print] : (For two at a time only.) Prints comparison data.
Check aflow --readme=compare

aflow --compare_structure=POSCAR_1,POSCAR_2 [--np=xx] [--print]
Compares POSCAR_1 to POSCAR_2 (no requirement on type of atoms,
just stoichiometry ratios). POSCARs must be in the same
directory you are running in.
[--np=xx] : Number of processors (default 8). Algorithm is thread-friendly.
[--print] : (For two at a time only.) Prints comparison data.
Check aflow --readme=compare

aflow --compare_material_directory|--compare_material_dir [-D "PATH"] [--np=xx]
Determines the unique materials (same atomic species)
within a given directory. Returns a JSON and TXT file with the
results. Default Directory: "."
[-D "PATH"] : User can specify a specific directory to compare.
Output will be placed there also.
[--np=xx] : Number of processors (default 8). Algorithm is thread-friendly.
Check aflow --readme=compare

```

`aflow --compare_structure_directory|--compare_structure_dir [-D "PATH"] [--np=xx]`

Determines the unique structure prototypes within a given directory.

Returns a JSON and TXT file with the results. Default Directory: "."

`[-D "PATH"]` : User can specify a specific directory to compare.

Output will be placed there also.

`[--np=xx]` : Number of processors (default 8). Algorithm is thread-friendly.

Check `aflow --readme=compare`

`aflow --convex_hull=|--chull --alloy=MnPdPt[,AlCuZn,...] [chull_options] [--path=[DIRECTORY]]`

Queries the AFLOW API for relevant entries (see `--load_library`), calculates the convex hull, and returns the information as a PDF (default, see `--output`).

`--chull` : Necessary flag for entering mode for calculating convex hull.

`--alloy` : Necessary argument, specifies system. This code is not dimension specific, i.e., you can calculate the convex hull for any n-ary system.

`[--path=[DIRECTORY]]` : Optional argument, specify the directory for the output.

Default is ".".

`[--usage]` : Returns usage commands and options.

`[--output=|--o=latex|pdf|json|text]` : Select the output format. Latex/PDF are the same (.pdf).

JSON and plain text have the following extensions: .json and .txt.

Default is Latex/PDF.

`[--image_only|--imageonly|--image]` : Latex/PDF output mode only. Similar to `--document_only`, but the image dimensions are not necessarily for a standard page. Preferred option for importing into papers/presentations.

`[--no_document|--nodocument|--no_doc|--nodoc|--full_page_image|--fullpageimage]` :

Latex/PDF output mode only. Exclude the report from the output. Differs from image only in that the convex hull illustration will be formatted for a standard page (8.5x11in).

This is good when generating lots of images that need to be the same size.

`[--document_only|--documentonly|--doc_only|--doonly|--doc]` : Latex/PDF output mode only.

Exclude convex hull illustration. This is the default for quaternary systems and above.

`[--stability_criterion=|--stabilitycriterion=|--stable_criterion=|--scriterion=|--sc=aflow:bb0d45ab555b`

Calculates the stability criterion of the ground state structure. Will return a warning if structure is not a ground state one. It removes the point from the hull, calculates the new hull, and calculates the distance of this point from below/above (Hf/Ts) hull.

`[--equilibrium_phases|--equilibriumphases|--ep]` : For each ground state structure, print the equilibrium phases that contain it. Very useful for determining which structures are in N-phase equilibrium. In essence, it is the text-version of the convex hull.

`[--keep_log|--keeplog|--log]` : Prints a log file of relevant output.

LOADING OPTIONS:

`[--load_library=|--loadlibrary=|--ll=icsd|lib1|lib2|lib3]` : Specify libraries from which to load. Default: icsd, lib2, and lib3.

`[--neglect=|--ban=aflow:bb0d45ab555bc208,aflow:fb9eaa58604ce774]` :

Ban specific entries from the convex hull calculation, done by AUID.

`[--see_neglect|--seeneglect|--sn]` : Show why entries were neglected.

`[--remove_extreme_points=|--removeextremepoints=|--remove_extrema=|--removeextrema=|--rep=-1000]` :

Exclude points based on Hf/Ts (floor/ceiling). Units are meV/atom / K.

`[--entropic_temperature|--entropictemperature|--entroptemp]` : Calculate the Ts convex hull (upper-half). Default is Hf hull (lower-half).

`[--ldau]` : Only include points with LDAU. Hulls cannot contain mix LDAU character.

`[--force_output|--forceoutput|--fo]` : By default, the convex hull calculation will quit if it does not identify any ground state structures. This flag will override the program exit and still produce an output file.

`[--load_entries_entry_output|--loadentriesentryoutput|--leo]` : Get full output for all entries loaded from the AFLOW API.

`[--load_API|--load_api|--loadapi|--lapi|--api]` : Force loading entries from the API (default unless on `nietzsche`, `aflowlib`, or `habana`).

GENERAL PLOTTING OPTIONS:

`[--keep_tex|--keeptex|--tex]` : Latex/PDF output mode only. Will keep latex .tex and put it in `--path=[DIRECTORY]`.

`[--plot_offhull=|--plotoffhull=|--offhull=true|false]` : Plot off hull compounds. Will not plot unstable explicitly asked by `--unstable` flag. By default, is true for 2D systems, false for 3D systems.

`[--unstable]` : Plot unstable points (above/below 0 Hf/Ts).

`--filter_by_z=|--filterbyz=|--fz=200` : Latex/PDF output mode only. Similar to `--remove_extreme_points` but this will only remove points from the convex hull illustration, but still include them in the :
`--filter_by_distance=|--filterbydistance=|--f_dist=|--fdist=|--fd=200` : Latex/PDF output mode only. Similar to `--filter_by_z`, but will filter based on distance from convex hull, and not Hf/Ts.
`--include_endpoints|--includeendpoints` : Latex/PDF output mode only. Include endpoints in the convex hull illustration. Default is off.
`--no_color_gradient|--nocolorgradient|--nocg` : Latex/PDF output mode only. Turns off all colors.
`--labels=both|none|off|compound|prototype` : Latex/PDF output mode only. Labels for compounds on the convex hull illustration. Default is compound. You can change to prototype, both (compound::prototype), or none/off.
`--plot_reduced_composition=|--plotreducedcomposition=|--plot_reducedcomposition=|--rc=true|false` : Latex/PDF output mode only. Show reduced composition label. By default, this is false for 2D systems true for 3D systems.
`--labels_off_hull|--labelsoffhull` : Latex/PDF output mode only. Similar to `--labels`, but for non ground state structures.
`--meta_labels|--metalabels` : Latex/PDF output mode only. Include metadata in the labels on the convex hull illustration. This includes Hf, Ts, and distance to hull.
`--no_hyperlinks|--nohyperlinks` : Latex/PDF output mode only. Turn off all hyperlinks to websites.
`--no_links|--nolinks` : Latex/PDF output mode only. Turn off all links from the convex hull illustration to the report.
`--kill_all_links|--killalllinks|--kal` : Latex/PDF output mode only. Turn off all links/hyperlinks.
`--large_font|--largefont|--large|--lf` : Latex/PDF output mode only. Prints a "larger" font size for convex hull illustration.
`--helvetica_font|--helveticafont|--helvetica` : Latex/PDF output mode only. Change font to helvetica.
`--no_bold_labels|--noboldlabels|--nobl` : Latex/PDF output mode only. Turn off any bold font.
`--no_rotate_labels|--norotatelabels|--norl` : Latex/PDF output mode only. Labels are rotated for space (default). This flag turns that option off.
`--composition_header_report|--compositionheaderreport|--composition_header|--compositionheader` : Latex/PDF output mode only. In the report, will print out stoichiometry group headers (compounds) with fractional compositions. Default is to show compositions in reduced form (reduced by greatest common multiple).
`--compounds_column_report|--compoundscolumnreport|--compounds_column|--compoundscolumn|--compound_colu` : Latex/PDF output mode only. Include a column for the compound name. Compounds are grouped by stoichiometry but each prototype will have a different atom count.
`--small_banner|--smallbanner|--sb` : Latex/PDF output mode only. Does not include AFLOW logo, count and date on the convex hull illustration.
`--no_banner|--nobanner|--nob` : Latex/PDF output mode only. Turns off banner information.
`--no_aflow_logo|--noaflowlogo|--noal` : Latex/PDF output mode only. Prints text logo instead of PDF version logo.
`--latex_output|--latexoutput` : Latex/PDF output mode only. See full latex output. Good for troubleshooting.
`--latex_interactive|--latexinteractive` : Latex/PDF output mode only. Allows you to interact with latex as it compiles. Good for troubleshooting.

2D PLOTTING OPTIONS:

`--reverse_axis|--reverseaxis|--ra|--flip_axes|--flipaxes|--fa` : Latex/PDF 2D output only. Flips the top and bottom axes and reverse the order of the points.

3D PLOTTING OPTIONS:

`--no_color_bar|--nocolorbar|--nocb` : Latex/PDF output mode only. Hides color bar, but keeps colors.
`--no_heat_map|--noheatmap|--nohm` : Latex/PDF output mode only. Turns off colors from facets, but keeps points colored.
`--no_color_ternaries|--nocolorternaries|--nocts` : Latex/PDF 3D output mode only. Ternary labels are black (not white).
`--color_ternaries_yellow|--colorternariesyellow|--cty` : Latex/PDF 3D output mode only. Ternary labels are white by default. This makes them yellow.
`--light_contrast|--lightcontrast|--lcl` : Latex/PDF output mode only. Modifies the convex hull illustration color scheme to be lighter.
`--hull_drop_shadow|--hulldropshadow|--hull_dropshadow|--dropshadow` : Latex/PDF 3D output mode only. Changes facet lines to be thin, white lines with thicker black background. Default is off (thick black facet lines).
`--draw_all_facet_lines|--drawallfacetlines|--daf1` : Latex/PDF 3D output mode only. Facet lines are only drawn if they have not already been drawn (overlapping lines) by default. This flag will draw all facet lines. Helpful if manipulating .tex file.

```

aflow --corners | --corner < POSCAR
  Add to the POSCAR the corner and face atoms so it looks
  good when you plot the unit cell with rasmol/jmol and so on..
  This option is just for visualization and should not be used
  to transform POSCARS to run. You will get super-imposed atoms!

aflow --data[=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] < POSCAR
  Outputs to standard out basic data about the structure in the
  POSCAR input file. Output includes volume, a b c alpha beta gamma,
  reciprocal lattice, reciprocal lattice volume.
  This routine consists of symmetry consistency checks. If symmetry rules are broken,
  the code automatically changes tolerance and recalculates from the beginning.
  Tolerance:
    The tolerance is given in Angstroms. There are two preset tolerances:
      tight: minimum_interatomic_distance/100.0
      loose: minimum_interatomic_distance/10.0
    The default tolerance is the "tight" tolerance value.
    A value can also be specified. (Note, it must not be below the minimum
    interatomic distance, otherwise an error will be thrown.)
  Options:
    [--no_scan] : Will not perform tolerance scan.
    [--print=txt| =json] : Specify output format.
aflow --data1=rcut < POSCAR
  This is basically like cmp_str except that it works on 1 str.
  Slightly different data is given: No space groups, All bond lengths.
aflow --data2 < POSCAR
  Outputs to standard out basic data about the structure in the
  POSCAR input file. Similar to --data but with another style (SC).
aflow --debye=THERMO
  Usage: aflow --debye=THERMO[.bz2]
  Fits Debye temperature to heat capacity data in THERMO or THERMO.bz2 file as calculated using APL.
  Writes results to file debye_temperature.dat and plots results in debye_temperature.png
  Also calculates value for Debye temperature which best fits all heat capacity data in the range
  given in the THERMO file, and writes this value in the file debye_temperature.dat
  See Ascroft & Mermin, Solid State Physics, equation 23.26 for expression for heat capacity in terms of d
  The following files must reside in DIRECTORY as in the AFLOW written form or as .bz2 compressed files.
THERMO
aflow.in
  The aflow.in file is used to determine the number of atoms per unit cell.
  If the files are found in .bz2 form, the compressed files are preserved and temporary files are written and d
  The generation of the THERMO file is controlled by the command
  [AFLOW_APL]TP
  See README_APL, especially 3). for more details.

aflow --diff=POSCAR1,POSCAR2
  Gives the difference in energy of the two structures within the PARTIAL OCCUPATION force field.

aflow --disp=cutoff < POSCAR
  Outputs to standard out the displacement from each atom to all
  neighbors within a distance cutoff. Also gives which unit cell the
  neighbour is in.
aflow --dist=cutoff < POSCAR
  Outputs to standard out the distances for each atom to all
  neighbors within a distance cutoff. Also gives which unit cell the
  neighbour is in.

aflow --delta_kpoints=number < POSCAR [or --dkpoints=number | -dkpoints=number | -dk=number]
  Gives the k1,k2,k3 so that the delta_kappa is less-equal than "number".
  Input dK is the desired smallest distance between grid points in reciprocal space.
  The smallest mesh to satisfy this requirement is calculated.
  See also --kpoints.

aflow --edata[=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] < POSCAR
  Outputs to standard the information given by --data in addition

```

to lattice-, superlattice- and reciprocal-lattice types. This routine consists of symmetry consistency checks. If symmetry rules are broken, the code automatically changes tolerance and recalculates from the beginning.

Tolerance:
 The tolerance is given in Angstroms. There are two preset tolerances:
 tight: minimum_interatomic_distance/100.0
 loose: minimum_interatomic_distance/10.0
 The default tolerance is the "tight" tolerance value.
 A value can also be specified. (Note, it must not be below the minimum interatomic distance, otherwise an error will be thrown.)

Magnetic analysis:
 The magnetic tag specifies the magnetic moment on each of the atoms. The two input formats are a comma separated string of the magnetic moments - in the same order as the input file, the location of the INCAR, or the location of the OUTCAR, which is then read and the magnetic moment for each atom is found (must be a static calculation, otherwise this information isn't specified). Works for both collinear and non-collinear systems (AFLOW autodetects collinear/non-collinear based on the input).

Options:
 [--no_scan] : Will not perform tolerance scan.
 [--print=txt| =json] : Specify output format.
 [--mag|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]]: Perform symmetry analysis of spin system, where

aflow --edos ispin
 ispin(1=nonspin,2=spin), Outputs to standard out the DOS, format:
 Energy DOSup DOSdown
 The integrated DOS is not outputted since it can be calculated from the energy bins and DOS. Needs DOSCAR and POSCAR files. DOS for spin down is given in (negative) sign. See also --kband

aflow --edos ispin s d
 aflow --edos ispin d s
 See --edos. Outputs total DOS and DOS of orbital s and d. The sDOS or dDOS is per species as given in POSCAR. Output format:
 Energy DOSup DOSdown s_up_spec1 s_down_spec1...s_up_specN s_down_specN d_up_spec1 d_down_spec1...d_up_specN

aflow --equivalent | --equiv | --iatoms[=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] [--mag|--magnetic|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]]: Calculate point/factor/space group and use them to label equivalent and inequivalent atoms. On the output, each structure atom has number_label_of_atom[equivalent_to_label]* (* if inequivalent). (SC1107).
 This routine consists of symmetry consistency checks. If symmetry rules are broken, the code automatically changes tolerance and recalculates from the beginning.

Tolerance:
 The tolerance is given in Angstroms. There are two preset tolerances:
 tight: minimum_interatomic_distance/100.0
 loose: minimum_interatomic_distance/10.0
 The default tolerance is the "tight" tolerance value.
 A value can also be specified. (Note, it must not be below the minimum interatomic distance, otherwise an error will be thrown.)

Magnetic analysis:
 The magnetic tag specifies the magnetic moment on each of the atoms. The two input formats are a comma separated string of the magnetic moments - in the same order as the input file, the location of the INCAR, or the location of the OUTCAR, which is then read and the magnetic moment for each atom is found (must be a static calculation, otherwise this information isn't specified). Works for both collinear and non-collinear systems (AFLOW autodetects collinear/non-collinear based on the input).

Options:
 [--no_scan] : Will not perform tolerance scan.
 [--print=txt| =json] : Specify output format.
 [--mag|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]]: Perform symmetry analysis of spin system, where

aflow --eigcurv=bands_directory
 Perform a graph decomposition of the BZ paths found in a BANDS calculation into component vertices and edges. These are then organized into a data structure that contains information on the branching associated with each edge in the graph. Once this is in place, short walks are performed along these edges and their branches

the purpose of detecting maxima and minima in the band structure, as well as their corresponding curvatures. The curvatures are currently calculated using the 5 point central difference approximation, $f''(f_3) = (-f_1 + 16f_2 - 30f_3 + 16f_4 - f_5)/(12h^2)$ which has an $O(4)$ error. Note that the curvatures are not reliable, given that angular information is lost in the vicinity of the vertices.

```
aflow --effective-mass | --em directory_name
Calculate the carrier effective masses.
If the data is generated by aflow, DOSCAR.static, POSCAR.static, and EIGENVAL.static are used.
Otherwise, DOSCAR, POSCAR, and EIGENVAL are used.
Output format:
  compound_name
  1. Band index:
    Carrier type:
    Carrier Spin:
    Effective masses along principle axes (m0):
    Number of equivalent valleys:
    DOS effective mass (m0):
    DOS electron effective mass (m0): {me_spin_up, me_spin_down}
    DOS hole effective mass (m0): {mh_spin_up, mh_spin_down}
```

```
aflow --enum | --multienum < POSCAR
Call MULTIENUM and convert a POSCAR with partial occupation into regular POSCARs.
```

```
aflow --enumsort | --multienumsort < POSCAR
Call MULTIENUM and convert a POSCAR with partial occupation into regular POSCARs, and sort POSCARs using
```

```
aflow --ewald[=eta] < POSCAR
Finds the electrostatic energy of the POSCAR file using the Ewald sum. Charges must be entered after each atom position.
E.g., Co 0.00 0.00 0.00 +2. This is easy to do using the --names option. Eta is a real space screening parameter. Setting eta<=0 or leaving it out will cause aflow to choose it automatically (and hopefully optimally). Eta->0 is no screening and the Ewald reciprocal term will be zero. Eta->inf is total screening and the Ewald real term will be zero. The Ewald sum itself should always be the same and eta will only affect the efficiency of the calculation.
```

```
aflow --extract_kpoints | --xkpoints aflow.in
Extract to stdout the kpoints embedded in the aflow.in file.
Useful for scripting (SC0209).
```

```
aflow --extract_incar | --xincar aflow.in
Extract to stdout the incar embedded in the aflow.in file.
Useful for scripting (SC0209).
```

```
aflow --extract_poscar | --xposcar aflow.in
Extract to stdout the poscar embedded in the aflow.in file.
Useful for scripting (SC0209).
```

```
aflow --extract_potcar | --xpotcar aflow.in
Extract to stdout the potcar embedded in the aflow.in file.
Useful for scripting (SC0209).
```

```
aflow --extract_symmetry | --xsymmetry aflow.in
Extract the symmetry (pgroup,fgroup,iatoms) from the poscar embedded in the aflow.in file. Must use this for aflow<2947 to fix the fgroup bug.
Useful for scripting (SC0209).
```

```
aflow --factorgroup[=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] [--screen_only] [--mag|--mag=]
Calculates factor group symmetry of the cell {R|t} and writes in the aflow.fgroup.out file. See documentation of aflow.
The point group is required for the factor group, therefore the aflow.pgroup.out file will be generated as well.
This routine consists of symmetry consistency checks. If symmetry rules are broken, the code automatically changes tolerance and recalculates from the beginning.
Tolerance:
```

The tolerance is given in Angstroms. There are two preset tolerances:

```

tight: minimum_interatomic_distance/100.0
loose: minimum_interatomic_distance/10.0

```

The default tolerance is the "tight" tolerance value.
A value can also be specified. (Note, it must not be below the minimum interatomic distance, otherwise an error will be thrown.)

Magnetic analysis:
The magnetic tag specifies the magnetic moment on each of the atoms. The two input formats are a comma separated string of the magnetic moments - in the same order as the input file, the location of the INCAR, or the location of the OUTCAR, which is then read and the magnetic moment for each atom is found (must be a static calculation, otherwise this information isn't specified). Works for both collinear and non-collinear systems (AFLOW autodetects collinear/non-collinear based on the input).

Options:

```

[--no_scan] : Will not perform tolerance scan.
[--print=txt| =json] : Specify output format.
[--screen_only] : Prints symmetry elements to string and does not write to a file.
[--mag|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]]: Perform symmetry analysis of spin system, where

```

`aflow --findsym[=tolerance_relative: default 1.0e-3] < POSCAR`
Runs and prints out the output of the program findsym

`aflow --findsym_print[=tolerance_relative: default 1.0e-3] < POSCAR`
This prints out the input file structure for findsym.

`aflow --findsymSG[=tolerance_relative: default 1.0e-3] < POSCAR`
This prints out the space group without bothering with wrap ups. findsym must be accessible on the path !

`aflow --findsymSG_label[=tolerance_relative: default 1.0e-3] < POSCAR`
Same as above but prints ONLY the space group name.

`aflow --findsymSG_number[=tolerance_relative: default 1.0e-3] < POSCAR`
Same as above but prints ONLY the space group number.

`aflow --fix_bands POSCAR KPOINTS.bands.old EIGENVAL.bands.old KPOINTS.bands.new EIGENVAL.bands.new`
Resample the PATH following the standard definition of the POSCAR and the standard path.
If there are not nearby points, it will ask to redo the nbands step.

`aflow --aflow-sym|--AFLOW-SYM|--AFLOWSYM|--aflowSYM|--afloWSym|--full_symmetry|--full_sym|--fullsym[=tolerance]`
Calculates the full symmetry of a crystal: point group lattice, point group crystal, factor group, space group, site point group, inequivalent/equivalent atoms, and the point group of the klattice. This routine calculates the full symmetry suite at a certain tolerance. If symmetry rules are broken, the code automatically changes tolerance and recalculates from the beginning.

Tolerance:
The tolerance is given in Angstroms. There are two preset tolerances:

```

tight: minimum_interatomic_distance/100.0
loose: minimum_interatomic_distance/10.0

```

The default tolerance is the "tight" tolerance value.
A value can also be specified. (Note, it must not be below the minimum interatomic distance, otherwise an error will be thrown.)

Magnetic analysis:
The magnetic tag specifies the magnetic moment on each of the atoms. The two input formats are a comma separated string of the magnetic moments - in the same order as the input file, the location of the INCAR, or the location of the OUTCAR, which is then read and the magnetic moment for each atom is found (must be a static calculation, otherwise this information isn't specified). Works for both collinear and non-collinear systems (AFLOW autodetects collinear/non-collinear based on the input).

Options:

```

[--no_scan] : Will not perform tolerance scan.
[--print=txt| =json] : Specify output format.
[--screen_only] : Prints symmetry elements to string and does not write to a file.
[--mag|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]]: Perform symmetry analysis of spin system, where

```

`aflow --frac [-f,-d,--fract,--fractional,--direct] < POSCAR`
Outputs to standard out a new POSCAR file with

atom positions in direct (fractional) coordinates.

`aflow --getTEMP[s] [--runstat | --runbar | --refresh=X | --warning_beep=T | --warning_halt=T]`
 If available, the command outputs the hostname and temperatures of the machine. Useful to find hardware
 with `--runstat` the command continuously prints the temperature, refreshing every XX refresh seconds;
 with `--runbar` the command prints a bar with the temperature, refreshing every XX refresh seconds;
 with `--refresh=X` you can specify the refresh time (DEFAULT below)
 with `--warning_beep=T`, if the max temp goes beyond T(C, DEFAULT below), the command beeps the computer
 with `--warning_halt=T`, if the max temp goes beyond T(C, DEFAULT below), the command halts the computer (aflow
 DEFAULT VALUES in `.aflow.rc`
`AFLOW_CORE_TEMPERATURE_BEEP=56.0 // Celsius`
`AFLOW_CORE_TEMPERATURE_HALT=65.0 // Celsius, you need to run aflow as root to halt`
`AFLOW_CORE_TEMPERATURE_REFRESH=5.0 // seconds`

`aflow --gulp < POSCAR`
 Outputs to standard out a gulp formatted input file based on
 the POSCAR input file. The formatting is for a distance
 calculation in gulp. If you want atom names you must put
 them after each atom position in the POSCAR file (see `--names`).
 If any names are missing they are defaulted to H.

`aflow --hkl=h,k,l[,bond] < POSCAR`
 Returns the planar density and number of broken bonds per unit are
 along the Miller plane (h k l). The calculations returns also
 the bonds broken for each type and all types.
 Note: the indices are with respect to the unit cell, not the
 conventional cell, and the result is in #atoms/A² (scale included).
 BOND: (default BOND_DEF):
 If you specify a "bond", all the bonds shorter than
`bond*nearest_neighbour_bond` (between all types or combination
 of them) that are cut by the plane will contribute to the broken
 bond density Nb(#/AA).
 NOTE2: the 1st atom is taken to be the origin (all the other atoms
 are shifted).

`aflow --hkl_search[=khlmx[,bond[,step]]] < POSCAR`
 Returns the planar density and number of broken bonds per unit
 are along the Miller plane (h k l). The calculations returns
 also the bonds broken for each type and all types.
 Note: the indices are with respect to the unit cell, not the
 conventional cell, and the result is in #atoms/A² (scale included).
 Search from `-hmax<=h<=hmax, -kmax<=k<=kmax, -lmax<=l<=lmax`, where
 the boundaries are (-4,-4,-4) to (4,4,4) unless the HKLMAX is specified.
 HKLMAX:
 If you specify a `hklmx`, then the search is
`-hklmx<=h<=hklmx, -hklmx<=k<=hklmx, -hklmx<=l<=hklmx`

BOND: (default 1.3):
 If you specify a "bond", all the bonds shorter than
`bond*nearest_neighbour_bond` (between all types or combination
 of them) that are cut by the plane will contribute to the broken
 bond density Nb(#/AA).

STEP: (default 1):
 If you specify STEP then the update of h,k,l, is done in steps of step.

`aflow --hkl_search_simple[=cutoff[,bond[,khlmx[,step]]]] < POSCAR`
`aflow --hkl_search_complete[=cutoff[,bond[,khlmx[,step]]]] < POSCAR`
 Returns the planar density and number of broken bonds per unit are
 along the Miller plane (h k l). The calculations returns also the bonds
 broken for each type and all types.
 Note: the indices are with respect to the unit cell, not the
 conventional cell, and the result is in #atoms/A² (scale included).
 The simple search is fast but it might miss some planes!
 The complete search should not miss planes but can be very slow!

CUTOFF (default 1.3):
 If you specify a cutoff, the command will look for all the
 planes generated with triplets of atoms in a radius `cutoff*|a1+a2+a3|`.
 The algorithm to find the density of atoms is quite complex,

and it relies on a sum of four triangles (one triangle v_1, v_2, v_3 generated by hkl) plus three triangles generated by adding the v 's and subtracting one. This guarantees that we span 2 unit cells "rhombi" in independent directions. The algorithm also plots the number of atoms in the radius, so you can choose a decent cutoff'. For bcc/fcc cutoff=1.5 usually good. If you do not specify the cutoff, a value of 1.3 is take by default. Be careful with the cutoff ! The complexity grows as cutoff^3 !

BOND: (default 1.3):
 If you specify a "bond", all the bonds shorter than $\text{bond} * \text{nearest_neighbour_bond}$ (between all types or combination of them) that are cut by the plane will contribute to the broken bond density $N_b(\#/AA)$.
HKLMAX (default "dims" given by cutoff)
 If you specify a $hklmax$, then the search is $-hklmax \leq h \leq hklmax$, $-hklmax \leq k \leq hklmax$, $-hklmax \leq l \leq hklmax$

STEP (default simple: 1 complete 1/12)
 If you specify STEP then the update of h, k, l , is done in steps of step.

NOTE1: The algorithm will be implemented in aflow as automatic surfaces generation.
NOTE2: the 1st atom is taken to be the origin (all the other atoms are shifted).

aflow --hnf n < POSCAR
 Given a POSCAR and a volume 'n' (determinant of the Hermite normal form), it generates the set of supercells of POSCAR having volume n and being unique.
 In more details: it calculates the HNF matrices, the point group of the POSCAR lattice, then reduces the supercells by symmetry and print the unique POSCARs in aflow.in START/STOP format, ready for high-throughput calculation.
 If the volume is chosen negative, then all the supercells with HNF from 2 to -n will be generated with the same algorithm.

aflow --hnftol [eps] < PARTCAR
 Given a PARTCAR, it calculates the HNF size to get the right concentration spread of partial occupation within the tolerance. If eps is zero, negative or it is not given, aflow takes the tolerance from the PARTCAR, and if this is not available, it takes the default value. PARTCAR

```

PARTCAR of Ag8.8Cd4Zr3.2 // EXAMPLE
-191.600 0.02 // POSCAR scale and tolerance if>0)
5.76 5.76 5.76 90 90 90 // usual a1,a2,a2 or ABCCAR or WYCKCAR definitions)
8*1+1*0.7333 4*1 1*0.266+3*1 // needs keywords * and + to determine the species and the occupations)
Direct Partial // you need to specify Partial, the "P" is mandatory)
0.25 0.25 0.25 Ag
0.75 0.75 0.25 Ag
0.75 0.25 0.75 Ag
0.25 0.75 0.75 Ag
0.25 0.25 0.75 Ag
0.75 0.75 0.75 Ag
0.25 0.75 0.25 Ag
0.75 0.25 0.25 Ag
0.50 0.50 0.50 Ag
0.00 0.00 0.00 Cd
0.00 0.50 0.50 Cd
0.50 0.00 0.50 Cd
0.50 0.50 0.00 Cd
0.50 0.50 0.50 Zr
0.50 0.00 0.00 Zr
0.00 0.50 0.00 Zr
0.00 0.00 0.50 Zr

```

aflow --hnfcell < POSCAR
 Convert a POSCAR with partial occupation into regular POSCARs, and generate a LOG.POCC file in working
 Given a PARTCAR, it calculates the HNF size and generate supercells.
PARTCAR Example:
PARTCAR of Se0.75Sn1Te0.25 [Sn1Se0.75Te0.25]

```

1.0 -4 (0.001) //POSCAR scale and [HNF size if <0 (useful when you kn
0.0000000000000000 3.0268425330530309 3.0268425330530309
-3.0268425330530309 3.0268425330530309 0.0000000000000000

```

```

-3.0268425330530309    0.0000000000000000    3.0268425330530309
1*0.75 1*0.25 1 //different species mut be separated by blank space
Direct Partial
0.5000000000000000 0.5000000000000000 0.5000000000000000 Se
0.5000000000000000 0.5000000000000000 0.5000000000000000 Te
0.0000000000000000 0.0000000000000000 0.0000000000000000 Sn
Another PARTCAR Example:
Se1.75Sn2
1.0 -4
2.1393597183461157 -5.9785152916018083 0.0000000000000000
2.1393597183461157 5.9785152916018083 0.0000000000000000
0.0000000000000000 0.0000000000000000 4.2994759406996081
1*0.75+1 2
Direct P
0.6424479395478624 0.3575520604521376 0.7500000000000000 Se
0.3575520604521376 0.6424479395478624 0.2500000000000000 Se
0.8714903476790070 0.1285096523209930 0.7500000000000000 Sn
0.1285096523209930 0.8714903476790070 0.2500000000000000 Sn

```

```

aflow --icsd Pb Sn Se < ternary.icsd
aflow --icsd Sn 34 Pb < ternary.icsd
aflow --icsd 34 82 50 < ternary.icsd
Output to standard out in "ICSD-format" (the NIST's Inorganic
Crystal Structure Database) all Pb-Sn-Se ternary-compounds.
The arguments for --icsd can be the elements' symbol or atomic number.
The input is from stdin ICSD-format in this example is streamed in
from file "ternary.icsd".
"ICSD-format" is generated by exporting the entry (hit) in NIST's Inorganic
Crystal Structure Database in long format using "FindIt" software (the stand-alone ICSD browser).
Examples are /common/NIST/binary.icsd and /common/NIST/ternary.icsd
More information on ICSD: http://www.nist.gov/srd/nist84.htm
(2009, wahyu@alumni.duke.edu)
aflow --icsd_allessthan Ra < input.icsd
(see --icsd). Output compounds if ALL elements in a compound have Z<Z_Ra
aflow --icsd_allmorethan Ra < input.icsd
(see --icsd_allessthan).
aflow --icsd_basislessthan 20 < input.icsd
aflow --icsd_basismorethan 20 < input.icsd
(see --icsd). Output all compounds having number of basis atoms in the PRIMITIVE cell
lessthan or morethan the specified parameter.
aflow --icsd_chem MgB4 < input.icsd
(see icsd). Extract MgB4 compound.
aflow --icsd_cubic < ternary.icsd
aflow --icsd_triclinic < ternary.icsd
aflow --icsd_monoclinic < ternary.icsd
aflow --icsd_orthorhombic < ternary.icsd
aflow --icsd_tetragonal < ternary.icsd
aflow --icsd_rhombohedral < ternary.icsd
aflow --icsd_trigonal < ternary.icsd
aflow --icsd_hexagonal < ternary.icsd
aflow --icsd_cubic --icsd_orthorhombic < ternary.icsd
(see --icsd). Output all compounds which belong to the specified system.
If multiple options, the relational is "OR".
aflow --icsd_tri < input.icsd
aflow --icsd_mcl < input.icsd
aflow --icsd_mclc < input.icsd
aflow --icsd_orc < input.icsd
aflow --icsd_orcc < input.icsd
aflow --icsd_orcf < input.icsd
aflow --icsd_orci < input.icsd
aflow --icsd_tet < input.icsd
aflow --icsd_bct < input.icsd
aflow --icsd_rhl < input.icsd
aflow --icsd_hex < input.icsd
aflow --icsd_cub < input.icsd

```

```

aflow --icsd_fcc < input.icsd
aflow --icsd_bcc < input.icsd
    (see --icsd). Output compounds which belong to the specified lattice type:
    tri (trigonal)
mcl (monoclinic)
mclc (c-centered monoclinic)
orc (orthorhombic)
orcc (c-centered orthorhombic)
orcf (face-centered orthorhombic)
orci (body-centered orthorhombic)
tet (tetragonal)
bct (body-centered tetragonal)
rhl (rhombohedral/trigonal)
hex (hexagonal)
cub (simple cubic)
fcc (face-centered cubic)
bcc (body-centered cubic)
aflow --icsd_denslessthan 4.5 < ternary.icsd
aflow --icsd_densmorethan 5.5 < ternary.icsd
aflow --icsd_densmorethan 4.5 --icsd_denslessthan 8.3 < ternary.icsd
    (see --icsd). Output all compounds with density (in g/cm3) lessthan and/or morethan
    the specified argument.
aflow --icsd_id 11120 < input.icsd
    (see --icsd). Extract compound with entry ID number 11120 of the original ICSD database.
aflow --icsd_lessthan Pb < binary.icsd
aflow --icsd_lessthan 82 < binary.icsd
    (see --icsd). Output all compounds in binary.icsd that contains AT LEAST
    ONE element with atomic number Z<Z_Pb (Z<82).
aflow --icsd_listmetals
    Output to stdout a list of metallic elements used in the --icsd_removemetals option.
aflow --icsd_makelabel < input.icsd
    Output to stdout a list of labels compatible to generate aflow.in from icsd database.
    The label uses the following format: chemicalformula_ICSD_entrynumber
    The chemicalformula is ascendingly sorted based on the element symbol and the
    concentration must be explicitly included eventhough it is 1.
    The entrynumber is the entry number of the compound in the ICSD database.
    e.g. Zn1Zr1_ICSD_106235
aflow --icsd_morethan Pb < binary.icsd
aflow --icsd_morethan 82 < binary.icsd
aflow --icsd_morethan Pt --icsd_lessthan Hg < binary.icsd
aflow --icsd_morethan 78 --icsd_lessthan 80 < binary.icsd
    (see --icsd_lessthan and --icsd).
aflow --icsd_n_ary 2 < input.icsd
    (see --icsd). Output all binary compounds
aflow --icsd_n_ary 3 < input.icsd
    (see --icsd). Output all ternary compounds. and so on
aflow --icsd_nobrokenbasis < input.icsd
    (see --icsd). Some compounds reported in the ICSD database have missing wyckoff position
    of some of the constituents. This option extracts only the ones with complete wyckoff pos.
aflow --icsd_nopartialocc < input.icsd
    (see --icsd). Output only compounds with full occupancies
aflow --icsd_proto #Nspecies1 #Nspecies2 #Nspecies3 ... < input.icsd
aflow --icsd_proto 2 1 7 < input.icsd
    (see icsd). Output all ternary compounds with general chemical formula A2BC7, AB2C7, A7B2C, and all
    other possible cyclic permutations.
aflow --icsd_remove Pu < input.icsd
    (see --icsd). Remove compounds which contain Pu.
aflow --icsd_removemetals < input.icsd
    (see --icsd). Remove compounds that are composed ENTIRELY by metallic elements.
    Metallic elements:
    Alkali metals, Alkaline earth metals, Transition metals,
    Lanthanoids, Actinoids, Other metals (Al, Ga, In, Sn, Tl, Pb, Bi).
    To list all metallic elements used in this option, use --icsd_listmetals
aflow --icsd_sg 62 < binary.icsd
aflow --icsd_sglessthan 62 < binary.icsd

```

```

aflow --icsd_sgmorethan 45 < binary.icsd
aflow --icsd_sgless than 194 --icsd_sgless than 231 < binary.icsd
    (see --icsd). Output all compounds where the space group number satisfies the
    specified arguments.
aflow --icsd_unique < input.icsd
    (see --icsd). Discard redundant compounds based on SG# AND ChemicalFormula

aflow --icsd_check_raw > output.dat
Checks the validity of the current vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/RAW/ dirs and files
(POSCAR KPOINTS DOSCAR and EIGENVAL)
with respect to vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/LIB/
output.dat will contain:
vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/RAW/./././ OK
vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/RAW/./././ OK
vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/RAW/./././ ERROR file1 file2 ...
vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/RAW/./././ NotInLIB
vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/RAW/./././ NotInLIB
vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/LIB/./././ NotInRAW
vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD)/LIB/./././ NotInRAW
and so on.
aflow --icsd_check_raw 0 > output.dat
same as the default --icsd_check_raw
aflow --icsd_check_raw 1 > output.dat
use the list of RAW and LIB dirs that have been compiled in aflow
i.e. XHOST.vGlobal.at(X) (ICSD_LIB); // aflow_data_calculated.cpp
    XHOST.vGlobal.at(X) (ICSD_RAW); // aflow_data_calculated.cpp

aflow --icsd2aflowin < input.icsd
    Convert input.icsd (ICSD-format) to aflow.in
aflow --icsd2poscar < input.icsd
    Write output to stdout POSCAR from stdin ICSD-format
aflow --icsd2proto < input.icsd
    Write output to stdout in ICSD_PROTO-format from stdin ICSD-format
aflow --icsd2wyck < ternary.icsd
    Write output to file WYCKCAR (WYCKCAR-format) from stdin ICSD-format
    in this example is streamed in from file "ternary.icsd".
    If the input contains multiple compounds, a subfolder will
    be created for each compound using the following folder name template:
        CompoundFormula_ICSD_ICSDid/
    The ICSDid is the compound id in the original ICSD database.
    ONLY compounds with elements having full occupation (sof=1) will be processed.
aflow --icsd2wyck --sof < ternary.icsd
    (see --icsd2wyck). In this case, all compounds will be processed.
    If partial occupation is detected, the sof will be written as part of
    the label of the element and WARNING will be written in the title and cerr.
aflow --icsdproto2aflowin < input.proto
    Convert ICSD-PROTOTYPE-format to aflow.in
    (2007-2011, wahyu@alumni.duke.edu)

aflow --identical < POSCAR
    Makes all the atoms identical. Useful to find the properties of
    the superlattice.
aflow --incell < POSCAR
    Outputs to standard out a POSCAR file with all
    atoms mapped to their images within the unit cell.
aflow --incompact < POSCAR
    Similar to --incell
    Outputs to standard out a POSCAR file where all the
    atoms are mapped through the unit and neighbours cells
    to minimize the shortest possible bond with an adjacent atom
    This option is very useful if you run big and complicate
    molecules where atoms exit of the unit cell and you have
    problems understanding where they are because visualization
    packages do not show bonds anymore ...
    Anyway, it is easier to test than to describe. (SC 6 Aug 04)

```

aflow --insphere radius < POSCAR
 Similar to --incell and --incompact
 Outputs in XYZ format (rasmol standard), the
 positions of all the atoms inside a sphere of radius
 "radius" centered in the origin.
 If you want another origin, you must shift the atoms first.
 The radius is in cartesian format. (SC 6 Aug 04)

aflow --intpol=file1,file2,nimages,nearest_image_flag
 Creates nimages image POSCAR files by interpolating linearly
 between structures in file1 and file2. File1 and file2
 should both be POSCAR like files with corresponding atoms.
 The nearest_image_flag is set to 'e' for exact interpolation
 and 'n' or 'N' for nearest-image interpolation. Exact
 interpolation means that all the positions are taken exactly
 as they are input. Nearest-image interpolation means that
 the interpolation between two corresponding atoms in file1
 and file2 is actually done between the atom in file1 and
 the nearest image of the corresponding atom in file2. This
 is useful, e.g., if the positions (in direct coordinates)
 are 0.01 (file1) and 0.99 (file2). These are far apart in
 exact interpolation but very close in nearest-image
 interpolation. The POSCAR output files are numbered and
 output to the present directory. The code also creates
 numbered subdirectories and copies the appropriate POSCAR
 files into those. This option makes it easy to set up a rubber
 band calculation in vasp.

aflow --inwignerseitz [--inws] < POSCAR
 Outputs to standard out a POSCAR file where all the atoms are
 mapped to their images in the Wigner-Seitz cell. (SC 10Jan04)

aflow --inflate_lattice=coefficient | --ilattice coefficient=coefficient < POSCAR
 Inflate/deflate POSCAR lattice of coefficient.

aflow --inflate_volume=coefficient | --ivolume coefficient=coefficient < POSCAR
 Inflate/deflate POSCAR lattice of coefficient.

aflow --kpath [--grid=XX] < POSCAR
 Outputs to stdout a KPOINTS in line mode useful for bandstructure
 calculations. The k-points path is chosen to cover all possible
 pair combinations of special k-points in the irreducible BZ (IRBZ).
 a special k-point is either:
 (1) a center of a face in IRBZ, or
 (2) a corner of IRBZ, or
 (3) a midpoint (say M) of a line-edge (say AB) IF the Gamma-M-A makes a 90 degrees angle.
 The coordinates are given in fractionals of the k-vectors, in which
 derived from the "standard" lattice used in AFLW.
 To find out the standard lattice of the POSCAR, use --sp option.
 Optional: --grid=XX specifies the density of the grid, if not specified it takes --grid=16.
 NOTE. If grid is negative real number, e.g. --grid=-0.01, then its value is taken
 as the deltaK during the path which is constructed as uniformly as possible in the k-space.
 The option works both for VASP, QE style mode inputs (ABINIT and AIMS in the future).
 Example: aflow --proto=3:Ag:Zr [--qe | --abinit | --aims] | aflow --kpath=-0.01

aflow --kpoints=KDENS [or --kppra=KDENS,-k=KDENS] < POSCAR
 This function prints a set of Monkhorst-Pack kpoint mesh values
 (divisions along each reciprocal lattice parameter)
 based on the desired KDENS density and the
 lattice parameters. The values assure the most even
 distribution of KDENS along the lattice params consistent
 with the kpt density. See --delta_kpoints

aflow --jmol=[n1[,n2[,n3[,color[,true/false]]]]] < POSCAR
 Similar to --xyz. Starts jmol (must be available) with a file
 based on the POSCAR input file. If you want to use atom names
 you must put them after each atom position in the POSCAR
 file (see --names). If any names are missing they are defaulted to H.
 If no numbers are specified, aflow takes "1,1,1". (SC/Dec09)
 Background color may be chosen (e.g., white, brown, beige, etc...).

Default background color is white. The flag "true" specifies that the cif file is to be saved. Default is "false." (RHT/Mar11)

`aflow --jmolanimation | --animation | --print=gif [gif_file] < POSCAR`
 Generate animated gif file of rotated structure from POSCAR by using jmol and ImageMagick. If no output gif file name is given, the default name "aninamtion_" + process id + ".gif" will be used. The setting parameters of jmol cannot be changed by users currently. User must have valid DISPLAY for jmol to open its window and "Convert" command is used to generate multi-gif by combining gif files generated by jmol.

`aflow --join_strlist strlist1 strlist2`
 Outputs to standard out a sequence of structures in POSCAR format. The structures consist of those in strlist1 with the atoms from strlist2 added in. The positions are taken from strlist2, transformed into Cartesian coordinates, and then inserted into strlist1. All inserted atoms are added as new types. If one file has more structures than the other then the shorter file is padded with copies of the last structure. For more information on a strlist see --make_strlist.

`aflow --kband ispin`
 ispin(1=nospin,2=spin). Outputs to standard out Eband vs K for band structure plot. Needs EIGENVAL and KPOINTS files. Format:
 Kpoint upEband1 upEband2...upEbandN downEband1 downEband2...downEbandN
 The Kpoint column is constructed by cascading each KPOINT-segment specified in KPOINT file.

`aflow --latticereduction | -latreduction POSCAR`
 Lattice Reduction to Max Orthogonality (Minkowski) and then to Niggly Form. This procedure has been shown to give the best unit cell for the lattice in terms of Minkowski optimization and most symmetric representation. For the bcc and fcc it gives the standard, most symmetric, unit vectors. It also helps in the primitive cell serach, as it has been included by default inside. (SC0902).

`aflow --lattice_type | --lattice | --lattice_crystal < POSCAR`
 Returns the lattice type and the conventional lattice type of the CRYSTAL following the tables of Setyawayn-Curtarolo [<http://dx.doi.org/10.1016/j.commatsci.2010.05.010>].

`aflow --lattice_lattice_type | --lattice_lattice < POSCAR`
 Returns the lattice type and the conventional lattice type of the LATTICE following the tables of Setyawayn-Curtarolo [<http://dx.doi.org/10.1016/j.commatsci.2010.05.010>].

`aflow --latticehistogram < POSCAR`
 Outputs the lattice calculated respect to tolerance of histogram of symmetry.

`aflow --use_LOCK=XXX`
 Uses XXX instead of "LOCK" in freezing/searching/operating directories. The option is very useful for compounded calculations.

`aflow --ltcell=a11,a12,a13,a21,a22,a23,a31,a32,a33 < POSCAR`
`aflow --ltcell=a11,a22,a33 < POSCAR`
`aflow --ltcell=file < POSCAR`
 Outputs to standard out the linear tranform of the input POSCAR file. This simpy multiplies cell parameters and atom positions by the 3x3 matrix values. This can be used to rotate the cell, swap x and y coordinates, etc. This cannot create a supercell (see --supercell for that). The nine numbers must be separated spaces, and they form the nine elements a11,a12,a13,a21,a22,a23,a31,a32,a33 of the 3x3 supercell matrix, respectively. If you specify only 3 numbers, the other six are taken zero. If you use the "file" syntax, nine numbers are read from file. They can be on one or multiple lines. New algorithm by SC (aug07).
 The new lattice is (by column) (a1 a2 a3)*=LT*(a1 a2 a3).

`aflow --ltcellfv=v1,v2,v3,phi < POSCAR`
 Rotates the lattice vectors and atoms by angle phi (in degrees) around vector (v1,v2,v3). Outputs to standard out the new POSCAR. This can be used to rotate the cell, swap x and y coordinates, etc.

`aflow --magpara=directory | --magpara (./)`
 Output shows the magnetic momentum, cell of volume and spin polarization around Fermi level. If a directory is not specified, it loads "./"
 EXAMPLE: `aflow --magpara`
 MAGNETIC MOMENTUM CELL : 2.74234
 MAGNETIC MOMENTUM ATOM : 0.685586
 VOLUME CELL : 53.43
 VOLUME ATOM : 13.3575
 SPIN DECOMPOSITION : 0.032,1.447,1.447,-0.062
 POLARIZATION FERMI : 0.755068

`aflow --slab=h,k,l[#filled_layers[#vacuum layers]] < POSCAR`
 Produces a slab unit cell in POSCAR format. The normal of the slab is specified by the Miller indices of the orthogonal plane. Thus, make sure you know how the original POSCAR is specified. For example, (100) is not what you think for fcc primitive. Each point (hkl) in the reciprocal lattice corresponds to a set of lattice planes (hkl) in the real space lattice. The reciprocal lattice vector points in a direction normal to the real space planes.

`aflow --make_strlist OUTCAR XDATCAR`
 Outputs to standard out a sequence of structures in POSCAR format. The strlist file created contains a list of structures, each formatted exactly like a POSCAR file, with an empty line after all but the last one. The file must have no extra lines at the end or all routines that read it will crash. The lattice parameters are pulled out from the OUTCAR and the positions from XDATCAR. If OUTCAR has too few lattice parameters then copies of the last set are used for the remaining XDATCAR.

`aflow --maxatoms=N | --max_atoms=N | --atoms_max=N | --atomsmax=N < POSCAR`
 Limits the input POSCAR to N atoms (it writes an error if atoms>N). This is a useful option for online wrappers.

`aflow --minkowski_basis_reduction | --minkowski | --mink < POSCAR`
 Converts the unit cell with the Minkowski reduction
 This routine takes a set of basis vectors (that form a lattice) and reduces them so that they form the shortest possible basis. The reduction is performed so that each vector "a_i" is as close as possible to the origin while remaining in the affine plane which is defined by "a_j", "a_k" but shifted by "a_i", for any choice of even permutations of i,j,k in 1,2,3.
 See Lecture notes in computer science, ISSN 0302-974, ANTS - VI : algorithmic number theory, 2004, vol. 3076, pp. 338-357
 ISBN 3-540-22156-5
 Written by Gus Hart in F90, recoded by SC in C++ (Sep/08).
<http://www.farcaster.com/papers/sm-thesis/node6.html>

`aflow --miscibility | --mix strings ..`
 Tells if in the library the system (one of the strings) was found to be miscible, immiscible or unknown.
`aflow --mix AgMg`
 returns AgMg MISCIBILITY_SYSTEM_MISCIBLE

`aflow --mom < POSCAR`
 Gets the mass moments of the POSCAR file. Gets only the first (center of mass) at this point. The scale is used in the calculation.

`aflow --msi < POSCAR`
 Outputs to standard out a msi file based on the POSCAR input file. This can be used as input for cerius.
 In the functions that output the msi format I do something to put the lattice vectors in a form that cerius2 can read happily. It involves making the third lattice vector parallel

to Z, the second once in the YZ plane, and the letting the first lattice vector have all 3 components. It is confusing and probably done in a silly manner, but it seems to work. If you want atom names you must put them after each atom position in the POSCAR file (see `-names`). If any names are missing they are defaulted to H. For an msi file one needs the atomic numbers. These are coded into the program for most atoms. If you use atoms with atomic numbers over 86 or f-electron atoms (Lanthanides and Actinides) then you will have to increase the database. For the list of all coded atomic numbers see the constructor for the structure class in `structure.cc`.

```
aflow --names A1 A2 ... < POSCAR
Outputs to standard out a POSCAR file with names
A1,A2, ... after the atom positions. Each name
Ai is assigned to all atoms of type i.

aflow --natoms | --numatoms < POSCAR
Outputs to standard out the number of atoms in the POSCAR.
Useful for scripting. (SC0902).

aflow --nbondxx < POSCAR
Outputs to standard out the nearest neighbor distances between all types of atoms in the POSCAR.
Useful for scripting. (C01017).

aflow --nanoparticle radius distance < POSCAR
Starting from the input poscar, aflow prints a nanoparticle
POSCAR with the radius and appropriate lattice making the
particles as far as "at least distance". The origin of the
particle is taken to be point 0,0,0 in the unit cell.
You can translate the origin before with the command --setorigin (see below)
to an atom, a point in cartesian or fractional coordinates.
The radius and distance are in Angstroms and not scale
normalized (all scales are set to 1.0).
If not specified the parameters are taken to be 10A. (SC Apr/08)

aflow --ndata < POSCAR
Outputs to standard output the following normalized data:
a1 a2 a3 phi(a2,a3) phi(a3,a1) phi(a1,a2)
a1,a2,a3 are NORMALIZED over V1/3 and phi are in degrees.

aflow --niggli POSCAR
Converts the unit cell to the standardized Niggli form. The
form is unique (up to some signs, I think). The transformation
makes use of only the lattice vectors and does not depend on the
basis atoms. This will work on any cell, but it treats the given
cell as primitive, and it will not reduce the cell to primitive
if it is not primitive already. At present the algorithm seems to
hang if I force more than about 6 digits of accuracy so be aware that
small errors might be introduced (these can break symmetry!).

aflow --nn < POSCAR
Outputs to standard out the nearest neighbour distance.

aflow --noorderparameter < POSCAR
Outputs to standard out a POSCAR without all the order parameter
stuff in it. (SC0903).

aflow --nosd < POSCAR
Outputs to standard out a POSCAR without selective dynamics
formatting. Combined with the above --sd option this allows
easy movement between POSCAR files with and without selective
dynamics formatting.

aflow --numnames A1 A2 ... < POSCAR
Same as names except appends an increasing integer to
each different atom type. Starts counting at 1 again
for each new atom type.

aflow --nspecies | --numspecies < POSCAR
Outputs to standard out the number of species in the POSCAR.
Useful for scripting. (SC0902).
```

```

aflow --OrthoDefect < POSCAR
  Computes the orthogonality defect of the lattice of POSCAR.
  The orthogonality defect is defined as the ratio of the product of
  lattice vector magnitudes to the volume of the parallelepiped defined
  by the lattice vectors. So, for three orthogonal vectors, this ratio
  would be unity. (R. Taylor).

aflow --pdb < POSCAR
  Outputs to standard out a protein database (PDB) format file
  based on the POSCAR input file. This can be used as input
  for many viewing programs. Note that I don't really know
  anything about PDB but it seems to be an annoyingly column
  formatted. Therefore, all the widths I use must be kept as is.
  This means that if we have width W for variable X, and prec P,
  then X takes up P+1 for the decimal part and decimal point,
  and we have only R=W-(P+1)-1=W-P-2 remaining digits (the -1 is
  because if X takes up all of W then you run into the previous
  field). So we have the constraints
  Cell vectors: W=9,P=3,R=4 => <=10^5
  Cell angles: W=7,P=2,R=3 => <=10^3 (which always works since
  angles are given as >=0 and <=360)
  Cartesian positions: W=12,8,P=3,R=7,3 => <=10^8,10^4 (>0)
  and <=10^7,10^3 (<0 since you need a space for the -- sign)
  This is all probably fine unless an atom makes it to more
  negative than -999.999.

aflow --pdos pdos.in PROOUT
  Writes the projected DOS for any desired combination of atoms,
  kpoints, bands, and lm values. The output consists of 6
  columns, spin up, down, up-down, and the cumulative DOS for
  each of those. Only up spin data is given for non
  spin-polarized calculations. The PDOS should look like
  an equivalent vasp output but will not be identical since
  vasp uses a different smearing method to get the PDOS.
  However, I think these PDOS are basically correct, at least
  qualitatively. To make this work you must
  -- Use slightly altered version of vasp
  -- set LORBIT=2 in INCAR
  -- Set ISYM=0 in INCAR
  -- Set RWIGS in INCAR (see below)
  The details explaining all this are given above in the --pocc
  section. The input file has the following format

  # Input for aflow --pdos.

  # These values you supply once.
  EMIN = --20.01477264 # default: 0.5eV below lowest energy.
  EMAX = 6.90250559 # default: 0.5ev above highest energy.
  NBINS = 300 # default: 300
  SMOOTH_SIGMA = 0.1 # Gaussian smoothing of the DOS.
  # default: 1 bin width.
  PRINT_PARAMS = 1 # 0=prints only data (easy to plot).
  # 1=prints all the input parameters.
  # default: 0

  # You can have as many cases as you want.
  # They are all added together.

  # case 1
  ATOMS = 1 # default: all atoms.
  KPOINTS = # default: all kpoints.
  BANDS = # default: all bands.
  LMVALUES = # default: all s,p,d,f.

  # case 1
  ATOMS = 2 # default: all atoms.

```

```

KPOINTS = 1 2 3 # default: all kpoints.
BANDS = 1 # default: all bands.
LMVALUES = 3 # default: all s,p,d,f.

```

All # denote comment lines. Each case is started when the token ATOMS is used. Following an ATOMS token, all KPOINTS, BANDS, LMVALUES tokens will apply to the atoms denoted in the preceding ATOMS token until the next ATOM token. The KPOINTS, BANDS, LMVALUES tokens can be left out in which case their default values will be used. You can have any number of cases. The above example will calculate a PDOS with projections onto atom 1 for all kpoints, bands, and s,p,d, and f (case 1) added to the projection onto atom 2 for kpoints 1-3, band 1, and the Pz orbital. I believe you can create any desired projections with this input file. The LMVALUES use the following correspondence between numerical input and orbitals projected.

```

Input number: 1 2 3 4 5 6 7 8 9 10 11
Orbitals:      S Py Pz Px Ptot Dxy Dyx Dz2 Dxz Dx2-y2 Dtot
Input number: 12 13 14 15 16 17 18 19 20
Orbitals:      F1 F2 F3 F4 F5 F6 F7 Ftot Tot

```

I have not done the work to figure out which of the standard f-orbital functions correspond to F1-F7.

This is only tested for version 4.4.5.

```

aflow --pearson_symbol | --pearson < POSCAR

```

Returns the Pearson symbol of the structure.

```

aflow --planedens dens2d.in CHGCAR

```

This finds the charge density in a plane. The input file dens2d.in has the form

```

D # Coordinates for following points (Direct/Cartesian)
scale # Scale factor - edges of plane get mult. by this (but not origin).
x y z # origin point
x y z # X axis
x y z # Y axis
Nx Ny # Number of X and Y grid points
Middle # Location for origin (Middle/Corner).
Ortho # Whether to use Y orthogonal to X (Ortho/Strict).

```

The output consists of 4 files, dens.[tot/diff/up/dn].out. Each has the same format, consisting of rows of density values, each row corresponding to a value along the X axis and each column to a value along the Y axis. The format can be read directly into Excel and easily into MatLab. This routine uses the same routine as --chgint to read in the CHGCAR and should work for the same versions of vasp (see --chgint for more information).

```

aflow --platon[=EQUAL | EXACT][,ang,d1,d2,d3] < POSCAR

```

This finds the space group. aflow is creating an output file, which is piped into a script platonSG, which uses the program platon.

Wraps input file for Platon ADDSYM package:

```

CALC ADDSYM (EQUAL) (EXACT) (ang d1 d2 d3)

```

where:

EQUAL - Search with all atom type treated as equivalent.

EXACT - All atoms should fit for given criteria.

ang - Angle criterium in search for metrical symmetry of the lattice (default 1.0 degree).

d1 - Distance criterium for coinciding atoms for non-inversion (pseudo)symmetry elements (default 0.25 Angstrom).

d2 - Distance criterium for coinciding atoms for (pseudo) inversion symmetry (default 0.25 Angstrom).

d3 - Distance criterium for coinciding atoms for (pseudo) translation symmetry (default 0.25 Angstrom).

The defaults are specified in `aflow_xatom.cpp` as

```

#define PLATON_TOLERANCE_ANGLE 1.0
#define PLATON_TOLERANCE_D1 0.25
#define PLATON_TOLERANCE_D2 0.25
#define PLATON_TOLERANCE_D3 0.25

```

To get the space group, type

```
aflow -platon < POSCAR | platonSG
```

To just get see the output file from `aflow`, type

```
aflow -platon < POSCAR
```

To check for errors and see output from `platon`, type

```
aflow -platon < POSCAR | platon -o
```

Note that the added flags above do not seem to work. To change tolerance create output file from `aflow`, and then add the four tolerances after `CALC ADDSYM` (on the same line). Then pipe this file to `platonSG`. `aflow` will use your atom labels if they are there. If you give no atom labels it will use defaults for each atom type, (these are He,Li,Be,B,C, and then W for all remaining atom types).
WARNING: If you have more than 6 atom types the W default will give the wrong space group. The equal flag may not work.
Atoms labeled with H do not get read by default in `platon`.
Do not use H labels.
SEE: <http://www.cryst.chem.uu.nl/platon/pl000401.html>
Note: it works with `platon.f` and `xdrvr.c > 51108`
`platon` must be accessible on the path !

```
aflow --platonSG[=EQUAL | EXACT][,ang,d1,d2,d3] < POSCAR
```

This prints out the space group without bothering with the wrap ups.
Note: it works with `platon.f` and `xdrvr.c > 51108`
`platon` must be accessible on the path !

```
aflow --platonSG_label[=EQUAL | EXACT][,ang,d1,d2,d3] < POSCAR
```

Same as above but prints ONLY the space group name.

```
aflow --platonSG_number[=EQUAL | EXACT][,ang,d1,d2,d3] < POSCAR
```

Same as above but prints ONLY the space group number.

```
aflow --plotband[=directory[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]
```

Generate bandstructure plot using `GNUPLOT`
In this function, the spin-polarized DOS plot can be generated!
The following files must reside in the directory:

```

DOSCAR.static
EIGENVAL.bands
KPOINTS.bands
OUTCAR.static

```

Defaults.
Directory: default = ./
Emin: The minimum value of the DOS plot. Default value -10
Emax: The maximum value of the DOS plot. Default value 10
DOSSCALE: The scale value of the DOS plot. Default value 1.2

Example:

```
aflow --plotband=/common/DATA/ZnO,-8,6 or
aflow --plotband=/common/DATA/ZnO,-8,6,1.2
```

```
aflow --plotband2[=directory[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]
```

In this function, the DOS is the sum of spin-up and spin-down!
The following files must reside in the directory

```

DOSCAR.static
EIGENVAL.bands
KPOINTS.bands
OUTCAR.static

```

Defaults.
Directory: default = ./
Emin: The minimum value of the DOS plot. Default value -10
Emax: The maximum value of the DOS plot. Default value 10
DOSSCALE: The scale value of the DOS plot. Default value 1.2

```
aflow --plotband_spinsplit[=directory[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]
```

This function generates spin-up and spin-down electronic band structure plots, respectively.
The following files must reside in the directory:

```
EIGENVAL.bands
```

```

    KPOINTS.bands
    OUTCAR.bands
Defaults.
Directory: default = ./
Emin: The minimum value of the DOS plot. Default value -10
Emax: The maximum value of the DOS plot. Default value 10
DOSSCALE: The scale value of the DOS plot. Default value 1.2
aflow --plotbanddos[=directory[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]
Generate bandstructure plot and DOS plot using GNUPLOT
This function equals to "aflow --plotband" & ""aflow --plotdos"
The following files must reside in the directory
DOSCAR.static
EIGENVAL.bands
KPOINTS.bands
OUTCAR.static
Defaults.
Directory: default = ./
Emin: The minimum value of the DOS plot. Default value -10
Emax: The maximum value of the DOS plot. Default value 10
DOSSCALE: The scale value of the DOS plot. Default value 1.2
aflow --plotdos[=directory[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]
Generate DOS plot using GNUPLOT
The following files must reside in the directory:
DOSCAR.static
OUTCAR.static
Defaults.
Directory: default = ./
Emin: The minimum value of the DOS plot. Default value -10
Emax: The maximum value of the DOS plot. Default value 10
DOSSCALE: The scale value of the DOS plot. Default value 1.2
Examples:
aflow --plotdos=/common/DATA/ZnO,-8,6
aflow --plotdos=/common/DATA/ZnO,-8,6,1.2
aflow --plotdosweb[=directory[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]
Generate DOS plot using GNUPLOT
The following files must reside in the directory
DOSCAR.static
Defaults.
Directory: default = ./
Emin: The minimum value of the DOS plot. Default value -10
Emax: The maximum value of the DOS plot. Default value 10
DOSSCALE: The scale value of the DOS plot. Default value 1.2

aflow --plotpedos[=directory[,number_atom[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]]
Generate PDOS plot using GNUPLOT
The following files must reside in the directory
DOSCAR.static
OUTCAR.static
Defaults.
Directory: default = ./
Number atom: default = 1
Emin: The minimum value of the DOS plot. Default value -10
Emax: The maximum value of the DOS plot. Default value 10
DOSSCALE: The scale value of the DOS plot. Default value 1.2
Example:
aflow --plotpedos=/common/DATA/ZnO,1
aflow --plotpedos=/common/DATA/ZnO,1,-8,6
aflow --plotpedos=/common/DATA/ZnO,1,-8,6,1.2
aflow --plotpedosall_nonquivalent[=directory[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]
Usage: aflow --plotpedosall_nonquivalent[=directory[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]
Generate PDOS plots of all the inequivalent atoms using GNUPLOT.
The following files must reside in the directory
DOSCAR.static
OUTCAR.static
POSCAR.relax1

```

Defaults.
Directory: default = ./
Emin: The minimum value of the DOS plot. Default value -10
Emax: The maximum value of the DOS plot. Default value 10
DOSSCALE: The scale value of the DOS plot. Default value 1.2
Use --keep=gnuplot (or --keep=GPL) to keep the GNU plot code.

`aflow --plotphonondispersion | --pphdis (--rcm | --meV | --THz | --hz) [--print=eps | --print=pdf | --print=png | --print=jpg | --print=gif]`
Usage: `aflow --pphdis --rcm /common/DATA/ZnO`
Generate Phonon dispersion, phonon DoS, F_vib, S_vib, and C_v plots in .eps form using
GNUPLOT. The flags [--rcm | --meV | --THz | --hz] specifies the units used in the graphs; --rcm is default
--rcm reciprocal centimeters
--meV mili-electron Volts
--THz teraHertz
--hz Hertz
The optional flags [--print=pdf | --print=png | --print=jpg | --print=gif] will trigger the conversion from
The following files must reside in DIRECTORY as in the AFLOW written form or as .bz2 compressed files
PDIS
PDOS
THERMO
If the files are found in .bz2 form, the compressed files are preserved and temporary files are written and deleted
The generation of these files is controlled by the commands
[AFLOW_APL]DC
[AFLOW_APL]DOS
[AFLOW_APL]TP
See README_APL, especially 3). for more details.

`aflow --pocc_dos[=directory[,T[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]]`
Generate density of states (DOS) plot at different temperature for a partial occupation structure.
If not specified, default directory is ./, temperature is 300K, DOS_Emin=-5 eV, DOS_Emax=5 eV, and DOSSCALE=1
Example: `aflow --pocc_dos=./,300`
More details about how to do this can be found in "README_AFLOW_POCC.TXT".

`aflow --pocc_mag[=directory[,T[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]]`
Calculate magnetic moment at different temperature for a partial occupation structure.
If not specified, default directory is ./, temperature is 300K, DOS_Emin=-5 eV, DOS_Emax=5 eV, and DOSSCALE=1
Example: `aflow --pocc_mag=./,300`
More details about how to do this can be found in "README_AFLOW_POCC.TXT".

`aflow --pocc_bandgap[=directory[,T[,DOS_Emin[,DOS_Emax[,DOSSCALE]]]]]`
Calculate band gap at different temperature for a partial occupation structure.
If not specified, default directory is ./, temperature is 300K, DOS_Emin=-5 eV, DOS_Emax=5 eV, and DOSSCALE=1
Example: `aflow --pocc_bandgap=./,300`
More details about how to do this can be found in "README_AFLOW_POCC.TXT".

`aflow --pocc PROOUT`
`aflow --pocc PROOUT`
Outputs occupations calculated from projections onto spherical harmonics for each ion for many combinations of L, M, bands, and kpoints. This works with version 445 and has not been tested on any other version (and probably won't work). To make this work you need to do the following
-- Use slightly altered version of vasp (see below)
-- set LORBIT=2 in INCAR (see below)
-- Set ISYM=0 in INCAR (see below)
-- Set RWIGS in INCAR (see below)
Here is why you need to do these things (feel free to skip this).
The --pocc option reads the PROOUT file, which is produced by running vasp with LORBIT=2 in the INCAR file. There is an annoying subtle point here. I am not sure I totally get this but here is my best understanding. The projection onto the spherical harmonics actually uses some atomic like radial functions (bessel functions). This means that for each spherical harmonic there are multiple states, corresponding to different atomic energy levels and radial functions.
Write the projection of band n, at kpt k, onto spherical harmonic with angular quantum numbers l,m and energy level e as Pnklme.

As compiled, the vasp code outputs projections Pnklm, summing over the e parameter. This makes each projection a sum of complex numbers, allowing some cancellations. However, for occupations, like those output in PROCAR (LORBIT=1) and OUTCAR, the summations over e are done with the squares of the Pnklme. This makes sense, since you want to add up probabilities, not amplitudes, to get an occupation. Unfortunately, from the output Pnklm one cannot reconstruct the Pnklme, so the output in PROOUT is not enough to reproduce the occupations. Therefore, I suggest the following. Recompile vasp with following modifications to sphpro.F

```
change line 252
      WRITE(IUP,'(9F12.6)') CSUM_PHASE
```

to

```
      write(IUP,'(9F12.6)') CSUM_ABS
```

CSUM_ABS is a complex variable but the imaginary part is zero. It is the squared amplitude for each projection. Note that this is a probability. You use it directly (do not square it) to get occupations. Note that the augmentations are added to this real number, and are therefore probabilities, not amplitudes. This makes sense when you look at the code in sphpro.F that calculates the augmentation portion. The augmentations can be <0, which I assume corresponds to reducing the probability of finding electrons. This makes me a little uncomfortable but I guess it is OK. At this point the aflow code assumes the above modification and uses the magnitude of the projections to calculate all occupations (not the magnitude squared).

Also, there is another subtle point with the kpoints. If you use symmetry certain sets of symmetry equivalent kpoints (a star) are represented by a single irreducible kpoint. We are used to ignoring this and simply weighting things associated with the irreducible kpoint appropriately to account for the whole star. However, the projections onto different orbitals are not the same for all the points in a star. For example, the kpoints (0.1,0,0), (0,0.1,0), (0,0,0.1) may all be in the same star in fcc, but states associated with them will project differently onto Px orbitals. Therefore, if you use irreducible kpoints you will get the wrong projections (it seems like you do get the right totals for S,P, and D, but I am not sure that is always the case). To be safe, don't use any symmetry -- i.e., set ISYM=0 in the INCAR file.

You must set RWIGS so that the code knows the radius of the spheres onto which it projects.

To make sure all is in order you can look at the total occupations for each ion in OUTCAR. These should match the Occupations vs. ION:LM values in the output of aflow (the last lines in the output). A more detailed check is to run vasp with LORBIT=1 and compare the PROCAR file with the output of aflow. I am not sure that this will all work the same way with PAW potentials.

```
aflow --pointgroup | --pgroup[=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] [--screen_only] <
```

Calculates the point group symmetry of the lattice {R} and writes it in the aflow.pgroup.out file. See documentation of aflow.

This routine consists of symmetry consistency checks. If symmetry rules are broken, the code automatically changes tolerance and recalculates from the beginning.

Tolerance:

The tolerance is given in Angstroms. There are two preset tolerances:

```
  tight: minimum_interatomic_distance/100.0
```

```
  loose: minimum_interatomic_distance/10.0
```

The default tolerance is the "tight" tolerance value.

A value can also be specified. (Note, it must not be below the minimum interatomic distance, otherwise an error will be thrown.)

Options:

```
  [--no_scan] : Will not perform tolerance scan.
```

```

    [--print=txt| =json] : Specify output format.
    [--screen_only] : Prints symmetry elements to string and does not write to a file.
aflow --pointgroupklattice | --pgroupk[=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] [--screen_
Calculates the point group symmetry of the klattice {K} and writes
it in the aflow.pgroupk.out file. See documentation of aflow.
This routine consists of symmetry consistency checks. If symmetry rules are broken,
the code automatically changes tolerance and recalculates from the beginning.
Tolerance:
    The tolerance is given in Angstroms. There are two preset tolerances:
        tight: minimum_interatomic_distance/100.0
        loose: minimum_interatomic_distance/10.0
    The default tolerance is the "tight" tolerance value.
    A value can also be specified. (Note, it must not be below the minimum
    interatomic distance, otherwise an error will be thrown.)
Options:
    [--no_scan] : Will not perform tolerance scan.
    [--print=txt| =json] : Specify output format.
    [--screen_only] : Prints symmetry elements to string and does not write to a file.
aflow --pointgroup_crystal | --pgroup_crystal | --pgroup_xtal[=tolerance| =tight| =loose] [--no_scan] [--prin
Calculate the point group symmetry of the crystal {R+atoms} and
writes it the aflow.pgroup_xtal.file.
This routine consists of symmetry consistency checks. If symmetry rules are broken,
the code automatically changes tolerance and recalculates from the beginning.
Tolerance:
    The tolerance is given in Angstroms. There are two preset tolerances:
        tight: minimum_interatomic_distance/100.0
        loose: minimum_interatomic_distance/10.0
    The default tolerance is the "tight" tolerance value.
    A value can also be specified. (Note, it must not be below the minimum
    interatomic distance, otherwise an error will be thrown.)
Magnetic analysis:
    The magnetic tag specifies the magnetic moment on each of the atoms. The
    two input formats are a comma separated string of the magnetic moments -
    in the same order as the input file, the location of the INCAR, or the location
    of the OUTCAR, which is then read and the magnetic moment for each atom is found
    (must be a static calculation, otherwise this information isn't specified).
    Works for both collinear and non-collinear systems (AFLOW autodetects
    collinear/non-collinear based on the input).
Options:
    [--no_scan] : Will not perform tolerance scan.
    [--print=txt| =json] : Specify output format.
    [--screen_only] : Prints symmetry elements to string and does not write to a file.
    [--mag|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]]: Perform symmetry analysis of spin system, where
aflow --pgl < POSCAR
Calculate the point group symmetry of the lattice {R} and
prints it to stdout.
aflow --pgx < POSCAR
Calculate the point group symmetry of the crystal {R+atoms} and
prints it to stdout.

aflow --pomass[=directory]
Outputs from the available POTCAR/OUTCAR the sum of POMASS.
aflow --pomass_atom[=directory]
Outputs from the available POTCAR/OUTCAR and POSCAR/CONTCAR the POMASS of the cell per atom.
aflow --pomass_cell[=directory]
Outputs from the available POTCAR/OUTCAR and POSCAR/CONTCAR the POMASS of the whole cell.

aflow --poscar < ABCCAR | WYCCAR
Converts the ABCCAR in POSCAR format.
ABCCAR is described as:
    TITLE
    SCALE (positive (rescaling) negative (volume))
    A B C ALPHA BETA GAMMA
    #specie0 #specie1 ....
    DIRECT (or CARTESIAN)

```

```

.. .. . specie0
.. .. . specie0
. . .
.. .. . specie1
.. .. . specie1
and so on. (Stefano Feb 2009)
Converts the WYCCAR IN POSCAR format.
WYCCAR is described as
TITLE
SCALE (positive (rescaling) negative (volume))
A B C ALPHA BETA GAMMA SG# [OPTION#]
#specie0 #specie1 ....
DIRECT (or CARTESIAN)
.. .. . specie0
.. .. . specie0
. . .
.. .. . specie1
.. .. . specie1
and so on. (Stefano Feb 2009)
The positions of the species will be used with the list of
symmetry operations (aflow_wyckoff.cpp) to generate all the atoms.

aflow --poscar2aflowin < POSCAR
Makes an "almost standard" aflow.in starting from the poscar.
Useful for scripting (SC100630).

aflow --poscar2enum < POSCAR
Convert a POSCAR into an input format for MULTIENUM (partial occupation).
This is useful for debugging MULTIENUM.

aflow --poscar2gulp < POSCAR
Convert a POSCAR into an input format for gulp program.

aflow --poscar2wyckoff < POSCAR
Outputs to Wyckoff positions using findsym.
Note that you may find more detailed information just using aflow --findsym < POSCAR if you can not und

aflow --prepare_chgcar_4_jmol|--prep4jmol=CHGCAR1[,CHGCAR2,...]

options:
[ --usage ]
[ --outcar=OUTCAR ]
[ --zip ]

Modify the header of a CHGCAR file so it can be read by Jmol.
Jmol requires species to be specified in the comment line of the CHGCAR file (first line).
This code requires an OUTCAR to read in the species.
Unless specified, it will first look for OUTCAR file in the directory of the CHGCAR, then it will
look in the directory from which the command is called.
The CHGCAR and OUTCAR files may be compressed, and the edited CHGCAR file can be recompressed
(if it was previously) by specifying the --zip command.

aflow --prim < POSCAR
Outputs to standard out a POSCAR file with a
primitive unit cell. In the primitive cell finding
function I look for a primitive cell by considering
as candidate cell vectors every possible triad of 3
vectors that can be made from the original cell vectors
or the basis vectors which translate the lattice onto
itself. I then take the triad with the smallest volume
to be the primitive cell. If there are multiple
candidates I take the ones with the largest projections
onto the original lattice vectors. See the routine
GetPrim.cc for more information. NOTE: the algorithm is by SC
different from the old DM aflow appraoce).

```

`aflow --primr | --fastprimitivecell | --fprim < POSCAR`
 Returns the primitive, Minkowski reduced cell, using a fast(er) routine. (R. Taylor).

`aflow [options] --proto=label*[:specieA*[:specieB*]..[:volumeA*[:volumeB*].. | :volume]] [--params=..... [---]`
 This command prints out a prototype of label="label" with speciesA,speciesB,... (non mandatory) of volumes per atom "volumeA, volumeB..." (not mandatory) or with standard volume per atom = "volume". Note that that `--proto label A B ...` is not supported anymore and you now need to use `--proto=label:A:B... et cetera.`

QUANTUM ESPRESSO output
 If you specify `--qe`, aflow will add QUANTUM ESPRESSO code to `aflow.in` (experimental).

ABINIT output
 If you specify `--abinit`, aflow will add ABINIT code to `aflow.in` (experimental).

AIMS output
 If you specify `--aims`, aflow will add AIMS code to `aflow.in` (experimental).

ANRL: AFLOW NAVAL RESEARCH LAB PROTOTYPE LIBRARY
 With respect to the article:
 Mehl et al., "The AFLOW Library of Crystallographic Prototype", <https://arxiv.org/abs/1607.02532>
 the code gives the possibility to define the parameters of the cell `--params=...`
 Some rhombohedral cells can be generated in hexagonal configurations (see the article).
 The choice is performed with the flag `--hex`.
 Check aflow `--readme=anrl`

`aflow --prototypermult | --PROTOTYPERMULT <POSCAR`
 Inputs 3 to 10 ary POSCARS, outputs the first alloy of prototype group, which is categorized by `--protoclassify`, from ICSD database.

`aflow --prototypes | --protos`
 Give the list of prototypes (file LIST.txt) from the DMQC-HTQC project (SC oct 08)

`aflow [--server=XXXXXXXX] --prototypes_icsd[=N] | --protos_icsd[=N]`
 Give the list of prototypes from the ICSD database included in the code.
 If N is passed, then it returns ONLY the N-ary compounds.
 If the ICSD_AFLOW_LIBRARY is not available, the code switches to downloading information from the `server=aflowlib.duke.edu` database.
 You can force downloading from different servers:
`--server=aflowlib.duke.edu`
`--server=materials.duke.edu`
`--server=default` (switches to `aflowlib.duke.edu`)

`aflow [--server=XXXXXXXX] [--vasp|--qe|--abinit|--aims] --proto_icsd=label`
 Returns the prototype labeled with the string label, from the ICSD database.
 Example: `aflow --proto_icsd=Ba1Ga4_ICSD_659522`
 Example: `aflow --aflowlib.duke.edu --proto_icsd=Ba1Ga4_ICSD_659522`
 If the ICSD_AFLOW_LIBRARY is not available, the code switches to downloading information from the `server=aflowlib.duke.edu` database.
 You can force downloading from different servers:
`--server=aflowlib.duke.edu`
`--server=materials.duke.edu`
`--server=default` (switches to `aflowlib.duke.edu`)
 The label can be of this type:
`--proto_icsd=ICSD_number.{ABC}[:specieA*[:specieB*]..[:volumeA*[:volumeB*].. | :volume]]"`
`--proto_icsd=label_ICSD_number"`

`aflow --qe`
 Transforms the POSCAR (or whichever is the input file format) to a QUANTUM ESPRESSO GEOM format.
 The geometrical file follows the QE convention (see `ibrav` in QE manual).

aflow --qvasp [--static] [-D directory]
 Given the current directory (or the one specified by -D), aflow produces the file aflow.qvasp.out containing thermodynamic information from relax2 CONTCAR/OSZICAR/OUTCAR (or .static if --static is specified).

aflow --rasmol[=n1[,n2[,n3]]] < POSCAR
 Similar to --xyz. Starts rasmol (must be available) with a file based on the POSCAR input file. If you want to use atom names you must put them after each atom position in the POSCAR file (see -names). If any names are missing they are defaulted to H. If no numbers are specified, aflow takes "1 1 1". (SC/OCT07,OCT14)

aflow --raytrace rtfile
 Outputs jpeg or mpeg pictures created by ray tracing. This is a rather elaborate routine. The basic idea is that it takes as input a list of structures (in POSCAR like format) and converts them into a dat file, inputs that into the ray tracing program tachyon, then takes the tga file output by tachyon and uses convert to make it a jpeg, then if needed takes all the jpegs and puts them together into and mpg using mpeg_encoder. Therefore the following programs must be installed and in your path: tachyon, convert, mpeg_encoder (only if making a movie). There is an input file rtfile which contains tokens setting characteristics of the calculation and the ray traced picture. This input file looks like the following:

```
# These set the input for the ray tracing program tachyon
CALCTYPE = 0 #0=strlist
INFILE = strlist # For CALCTYPE=0
OUTFILE = XXX # All output will have this prefix.
RESX = 600 # X pixels in images.
RESY = 500 # Y pixels in images.
ZOOM = 1.5 # Like a zoom lens.
ASPECTRATIO = 1 # Y height / X height.
ANTIALIASING = 0 # Something to do with extra accuracy.
RAYDEPTH = 12 # Number of reflections to keep in ray trace.
# x,y,z of camera location. Changes incrementally each frame
# from initial to final value. format is ciX cfX ciY cfY ciZ cfZ,
# where i,f denote initial and final, respectively.
# Default is displaced along -Y from structure center.
#CENTER = 1 1 -2 -2 2 4
#VIEWDIR = 1 0 0 # Direction of camera viewing.
#UPDIR = 0 0 1 # Up direction for picture.
BACKGROUND = 0.3 0.1 0.1 # Background lighting (all lights are R G B).
LIGHT = -20 -20 -20 0.01 1.0 1.0 1.0 # Spherical light source: center(3) radius(1) color(3)
LIGHT = 20 20 20 0.01 1.0 1.0 1.0 # Spherical light source: center(3) radius(1) color(3)
ATOMTEXTURE = 1 0.1 0.9 0.0 1.0 # atomtype ambient diffuse specular opacity
ATOMTEXTURE = 2 0.1 0.9 0.0 1.0 # atomtype ambient diffuse specular opacity
ATOMTEXTURE = 3 0.1 0.9 0.0 1.0 # atomtype ambient diffuse specular opacity
ATOMTEXTURE = 4 0.2 0.9 0.3 1.0 # atomtype ambient diffuse specular opacity
ATOMCOLOR = 1 1.0 0.75 0.3 # atomtype(1) color(3)
ATOMCOLOR = 2 0.2 0.4 1.0 # atomtype(1) color(3)
ATOMCOLOR = 3 1.0 1.0 1.0 # atomtype(1) color(3)
ATOMCOLOR = 4 1.0 0.0 0.0 # atomtype(1) color(3)
ATOMRAD = 1 0.6 # atomtype rad
ATOMRAD = 2 0.3 # atomtype rad
ATOMRAD = 3 1.0 # atomtype rad
ATOMRAD = 4 0.8 # atomtype rad
SHADING = mediumshade # fullshade,mediumshade,lowshade,lowestshade
# A supercell matrix given as 9 reals: a11 a12 a13 a21 a22 a23
# a31 a32 a33. Note that using this moves all atoms into images
# the unit cell.
# Default = 1 0 0 0 1 0 0 0 1
SUPERCELL = 1 0 0 0 1 0 0 0 1
# Rotation around x,y,z axis through structure_origin
```

```

# (counterclockwise). Rotation take place incrementally each
# frame. Rotation goes from initial to final value.
# format is riX ifX riY rfY riZ rfZ, where i,f denote initial
# and final, respectively.
# Default = 0 0 0 0 0 0
ROTATION = 0 0 0 0 -180 180
# The zero for the structure coordinates and lattice parameters.
# Rotations occur around this point.
# Default = First moment of atom positions.
STRUCTURE_ORIGIN = 0 0 0
PLANE = 1 # 0 for no plane, 1 for plane.
PLANECENTER = 0 0 -9.21016 # X Y Z for center of plane.
PLANENORMAL = 0 0 1 # Direction of plane normal.
PLANECOLOR = 1 1 1 # R G B for plane color.
PLANETEXTURE = 0.3 0.6 0.8 1.0 # ambient diffuse specular opacity for plane.

```

Tokens can be in any order, whitespace is ignored, and anything following a # on a line is ignored. The CENTER, VIEDDIR, and UPDIR depend on the structure and can be a pain to set by hand so they have fairly elaborate defaults that seem to usually work fine. The other values are not bad places to start. For more info about these check out the tachyon documentation.

For a single structure the output consists of XXX.dat, XXX.tga, XXX.jpg, XXX.enc, XXX.mpg. Having all these files is useful if some part of the program did not execute properly and you need to do steps by hand (tachyon XXX.dat produces XXX.tga, convert XXX.tga XXX.jpg produces XXX.jpg, and mpeg_encode XXX.enc produces XXX.mpg). For more than one structure the XXX.dat and XXX.tga files are erased to avoid clutter. The XXX.enc file is used for the mpeg_encode programs and is totally set in aflow and the user has not control over it from the rtf file. If you want to alter it after it is output go ahead but I know almost nothing about how it works. To make your own mpg from the jpg type: mpeg_encode XXX.enc
The tga, jpg files can be viewed with xv and the mpg with mpeg_play.XXX
Note: this routine was written by Dane and readapted by Stefano.
It might not work but require slightly modifications in the C++ source.

TACHYON WEB: <http://jedi.ks.uiuc.edu/~johns/raytracer/>
MPEG_ENCODER: http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_encode.html
CONVERT: <http://www.imagemagick.org/script/index.php>
LINUX: Debian 4.0r1 contains convert ("imagemagick") and the mpeg_encoder ("ucbmpeg").

aflow --rbanal nim nearest_image_flag

This is meant to allow you to analyse a rubber band calculation to get energy vs. distance along activation path. The distance is given a cumulative from the 00 to END images (a line integral), and also as the total distance from 00 and END for each image. The first is the activation path, the others are for double checking. The rubber band run must have had nim images. The distance between two images is defined to be the square root of the sum of all the squared distances between all the atoms in the two images. The distances depend on the setting of nearest_image_flag (see --intpol). The code works in two steps. First it gets energies and distances from the OSZICAR and POSCAR/CONTCAR files, respectively. This requires that it be run in the directory above all the image directories. The distances are obtained from POSCAR files for the first and last images, which are not actually calculated so now CONTCAR files exists. For intermediate images the POSCAR files are used. The energies are pulled from the OSZICAR files. First, these do not exist in the first and last image directories, so you must put them their yourself based on other runs (if you leave them out then no energies will be obtained but you can add the energies later, as described

below). Second, the output format for OSZICAR in a parallel vasp rubber band run is all messed up. Therefore, the energies are not E0 values (which are not the energy values of the images) but the free energy from the line just before the last E0. These numbers differ somewhat from E0 values. This whole output formatting problem changes with version of vasp -- I set it up for version 4.4.1 beta. It should also work with v 454. We should change it to use E0 as soon as possible. Once the distances and energies have been obtained then they are interpolated with a cubic spline curve. The spline interpolation assumes that the derivatives at the end points are 0. The E.vs.dist data and the spline interpolation are both output to standard out. If there is any problem with the energy data (e.g., you need to add end members) then you can fix it and run the spline interpolation again independently (see --spline). The spline interpolation is done on 50 points by default. This can be changed by setting DEFAULT_NPT_FOR_SPLINE in aflow_pflow_funcs.cpp.

aflow --rbdist POSCAR1 POSCAR2 n|N|e|E
 Gets the distance between two POSCAR files. Uses lattice params of POSCAR1. For use of n|N|e|E see --intpol.

aflow --rdf[=rmax[,nbins[,sigma]]] < POSCAR
 Get the radial distribution function (rdf) for a POSCAR file
 rmax: The radius out to which the rdf is calculated.
 nbins: The number of bins for the rdf.
 sigma: The sigma of the gaussian used to smear the rdf.
 This also finds the nearest-neighbor shells by looking at where the rdf has minima. The output gives the rdf and nn shells for each atom and for each type of neighbour, including a sum of all neighbour types. If you want to treat all atoms as the same (e.g., to look at the parent lattice) just set one type in the POSCAR file. You may need to play with sigma and the nbins to get atoms grouped together that you want in one shell but not to group atoms you want in different shells. One approach is to choose enough bins to make sure to distinguish every shell of interest and then increase sigma until you group desired atoms together.
 rmax default = 5
 nbins default = 25
 sigma default = 0 (zero does no smearing).

aflow --rdfcmp=rmax,nbins,sigma,nshmax,POSCAR1,POSCAR2
 Uses the radial distribution functions (rdf) for POSCAR 1 and 2 to assess how close the structures are. For each atom the rdf and radial shell function (rsf - this gives the coordination numbers for each shell) are found for each type. The shells are found by looking at the derivatives of the rdf. Then the atoms of the same types are compared and the rms errors in the rsf for POSCAR1 and POSCAR2 are computed. Then we step through the atoms of POSCAR1, matching them up with the atoms of POSCAR2, based on the minimum rms. When a POSCAR2 atom is matched it is not considered for later matches. The total rms based on these best matches is found by averaging all the individual atom RMS's. The total RMS and the rsf for the best matched atoms are output.
 rmax: The radius out to which the rdf is calculated.
 nbins: The number of bins for the rdf.
 sigma: The sigma of the gaussian used to smear the density.
 nshmax: The farthest possible shell used for computing RMS with the rsf (if <nshmax shells are found then fewer will be used).

I think this works but large changes in shape give rsf that are too different to compare exactly and the RMS!=0. However, looking at the output can make it clear how the shells are related. At this point the 2 structure must have the same number of atoms of each type. If you want to compare 2 POSCARs

where one is a multiple of the other in terms of atoms of each type you can create supercell (see --supercell) to make this function usable.

`aflow --revsg [#] [n] [l] [m]`
Constructs a POSCAR of a specified space group and with specified wyckoff positions
is the space group number
n multiplicity of wyckoff site (e.g., do position 4 g 1 twice)
l specifies multiplicity group (e.g., positions with multiplicity 2)
m position within multiplicity group (e.g., 2 would indicate the second position with multiplicity defined by l). (R. Taylor).

`aflow --rm_atom iatom < POSCAR`
Outputs to standard out a POSCAR where iatom has been removed.
iatom must be between 0 to N-1.

`aflow --rm_copies < POSCAR`
Outputs to standard out a POSCAR where only the first appearance of each cartesian position of the same type atom has been kept.
Useful if you have multiple copies of atoms at the same positions for some reason. Note. This routine does not compare equivalent lattice positions so you are not sure that equivalent atoms are not in the same place.

`aflow --rsm < POSCAR`
Outputs to standard out a Rasmol (rsm) format file based on the POSCAR input file. This output can be saved as .rsm file and is designed to be visualized in simpler version of Rasmol which is called RasTop program. It might work in Rasmol program directly. The format includes the plotting of the unit cell wireframe and spacefill 150 for the atomic radius. (WS Sept07)

`aflow --rsm --z z_type1 z_type2 ... z_typeNtypes < POSCAR`
Similar to `aflow -rsm`, with additional option of atom labeling based on the atomic number (z) of each type. If --z option is not used, the default is --z 1 2 3 ... Ntypes. (WS Sept07)

`aflow --sc | --std_conv | --standard_conventional | --sconv < POSCAR`
Output POSCAR in a standard conventional lattice
(use --sp | --std_prim | --standard_primitive | --sprim to output POSCAR in a standard primitive lattice) (WS & SC Nov09)
REF: Setyawan Curtarolo, DOI: 10.1016/j.commatsci.2010.05.010

`aflow --scale=s < POSCAR`
Outputs POSCAR file giving same volume as input but with scale = s.

`aflow --sd AA1 AA2 ... < POSCAR`
Outputs to standard out a POSCAR with selective dynamics formatting. AAi gives the selective dynamics setting for atoms of type i and has the form AAi=TTT,TTF,etc.. If there are fewer AAi than atom types then the remaining types are defaulted to TTT. A warning will be printed and the default value of TTT will be used if the AAi strings are shorter than three characters. (SC0907)

`aflow --setcm cm1 cm2 cm3 < POSCAR`
Sets to center of mass to (cm1,cm2,cm3) by shifting all the atom positions. Use --mom to check the shift was correct.

`aflow --setorigin r1 r2 r3 | atom# < POSCAR`
With r1 r2 r3, it sets the origin to r1,r2,r3 by shifting all the atom positions. Coordinates (r1,r2,r3) are considered fractional is the POSCAR is in fractional, and cartesian otherwise.
With atom#, it sets the origin to the position of atom# (atom# goes from 0 to size-1).

`aflow --sewald eta < POSCAR`
Finds the screened electrostatic energy of the POSCAR file using a real space sum. eta is now the screening length (ie, all coulomb interactions are multiplied by $\exp(-\eta * R)$. Charges must be entered after each atom position.
E.g., Co 0 0 0 +2.

`aflow --sgdata | --space_group_data[=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] [--no_scan]`
 Outputs the following space group information: space group number, space group labels (Hermann Mauguin, Hall, Schoenflies), Laue class, crystal class, ITC setting, ITC origin shift, general Wyckoff position, and the WYCCAR (similar to `--edata`).
 This routine consists of symmetry consistency checks. If symmetry rules are broken, the code automatically changes tolerance and recalculates from the beginning.

Tolerance:
 The tolerance is given in Angstroms. There are two preset tolerances:
 tight: minimum_interatomic_distance/100.0
 loose: minimum_interatomic_distance/10.0
 The default tolerance is the "tight" tolerance value.
 A value can also be specified. (Note, it must not be below the minimum interatomic distance, otherwise an error will be thrown.)

Magnetic analysis:
 The magnetic tag specifies the magnetic moment on each of the atoms. The two input formats are a comma separated string of the magnetic moments - in the same order as the input file, the location of the INCAR, or the location of the OUTCAR, which is then read and the magnetic moment for each atom is found (must be a static calculation, otherwise this information isn't specified).
 Works for both collinear and non-collinear systems (AFLOW autodetects collinear/non-collinear based on the input).

Options:
`--no_scan` : Will not perform tolerance scan.
`--print=txt| =json` : Specify output format.

`aflow --shell=ns,r1,r2,name,dens < POSCAR`
 Based on the structure in POSCAR, this outputs to standard out all points with ns neighbors in a shell define by inner radius r1 and outer radius r2 ($r1 < r2$). The shell is restricted to only consider atoms of type name (name=NONE ("") is unrestricted). The dens is the linear density of candidate points. So for example, to search for all sites with 4 oxygen around them between 1.8 and 2.2 Angstrom you could type
`aflow --shell=4,1.8,2.2,0,20`
 This would construct a 20x20x20 grid of puts in the cell and print out all points that met the shell criteria. The output gives all the points and their shells meeting the shell criteria. However, since many points are in the same shell it is convenient to find unique shell environments. Therefore, aflow reduces the complete list to keep only one representative point from each unique shell. The shells are distinguished by their centers of mass. The unique points and their shells are output as all unique points meeting the shell criteria. The representative point for each shell is the shell center of mass. This routine is great for finding open tetrahedral and octahedral sites where one might put an intercalant. This routine can take a while to run (usually a 20x20x20 mesh is enough, and might take a couple of minutes for a big cell - start small!).

`aflow --shift=Sx,Sy,Sz[,cCdD] < POSCAR`
 Outputs to standard out a POSCAR with all positions shifted by $S=(S_x, S_y, S_z)$. The shifted is added to the positions. The shift is assumed to be given in Cartesian coordinates unless you specify c or C for Cartesian or d or D for direct at the end.

`aflow --sitepointgroup | --agroup[=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] [--screen_only]`
 Calculates the site point group symmetry for every atom in the unit cell and writes it in the `aflow.agroup.out` file.
 This routine consists of symmetry consistency checks. If symmetry rules are broken, the code automatically changes tolerance and recalculates from the beginning.

Tolerance:
 The tolerance is given in Angstroms. There are two preset tolerances:
 tight: minimum_interatomic_distance/100.0
 loose: minimum_interatomic_distance/10.0
 The default tolerance is the "tight" tolerance value.
 A value can also be specified. (Note, it must not be below the minimum

interatomic distance, otherwise an error will be thrown.)

Magnetic analysis:
The magnetic tag specifies the magnetic moment on each of the atoms. The two input formats are a comma separated string of the magnetic moments - in the same order as the input file, the location of the INCAR, or the location of the OUTCAR, which is then read and the magnetic moment for each atom is found (must be a static calculation, otherwise this information isn't specified). Works for both collinear and non-collinear systems (AFLOW autodetects collinear/non-collinear based on the input).

Options:
[`--no_scan`] : Will not perform tolerance scan.
[`--print=txt| =json`] : Specify output format.
[`--screen_only`] : Prints symmetry elements to string and does not write to a file.
[`--mag|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]`]: Perform symmetry analysis of spin system, where

See documentation of `aflow`.

`aflow --spacegroup [=tolerance| =tight| =loose] [--no_scan] [--print=txt| =json] [--screen_only] [--radius] [`
Calculates space group symmetry of the cell {R|t+T} with translations as big as |T| and writes it in the `aflow.sgroup.out` file. See documentation of `aflow`.
Be careful because the size of the space group increases as the radius³ times the size of the factor group.
The point and factor groups are required for the space group, therefore the `aflow.pgroup.out` and `aflow.fgroup.out` files will be generated as well.
This routine consists of symmetry consistency checks. If symmetry rules are broken, the code automatically changes tolerance and recalculates from the beginning.

Tolerance:
The tolerance is given in Angstroms. There are two preset tolerances:
tight: `minimum_interatomic_distance/100.0`
loose: `minimum_interatomic_distance/10.0`
The default tolerance is the "tight" tolerance value.
A value can also be specified. (Note, it must not be below the minimum interatomic distance, otherwise an error will be thrown.)

Magnetic analysis:
The magnetic tag specifies the magnetic moment on each of the atoms. The two input formats are a comma separated string of the magnetic moments - in the same order as the input file, the location of the INCAR, or the location of the OUTCAR, which is then read and the magnetic moment for each atom is found (must be a static calculation, otherwise this information isn't specified). Works for both collinear and non-collinear systems (AFLOW autodetects collinear/non-collinear based on the input).

Options:
[`--no_scan`] : Will not perform tolerance scan.
[`--print=txt| =json`] : Specify output format.
[`--screen_only`] : Prints symmetry elements to string and does not write to a file.
[`--mag|--magnetic|--magmom=[m1,m2,...|INCAR|OUTCAR]`]: Perform symmetry analysis of spin system, where

`aflow --sqs=structure_type,atom_num,neighbour_num,sl_num_min,sl_num_max,A,B | --special-quasirandom-structure`
Generate special-quasirandom-structure (SQS)
structure: `bcc/fcc/hcp`
atom_num : maximum number of atoms in a cluster
neighbour_num : maximum number of nearest neighbour pairs in a cluster
sl_num_min : minimum number of the base structure in the superlattice
sl_num_max : maximum number of the base structure in the superlattice

To generate sqs bcc structure, one needs first to generate a 'cluster' file and a bcc superlattice file. Use the following commands to get sqs bcc from 5 to 8 atom/cell of AuCu alloy with energy calculated by using cluster expansion method with cluster containing up to 6th nearest neighbour pairs.
=====

1. Get cluster file. Number of atoms in a cluster is from 1 to 2 and distance of two atoms in a cluster is from the 1st to the 6th neighbour distance.
`aflow --cluster=bcc,1,2,1,6`

2. Get all supercell configuration. Here we get supercell with 8 atom. The first 5 is the smallest number of atoms in a supercell and the second one is the largest number.

```
aflow --superlattice=bcc,6,8
```

3. Get SQS. Check correlations of clusters with up to 2 atoms and 6th neighbour distance. You can give any two element names for the last two arguments.

```
aflow --sqs=bcc,2,6,5,8,Au,Cu
```

```
aflow --species < POSCAR
```

Outputs the species names, as taken from the 1st representative atom of each set.

```
aflow --spline npt < file
```

Outputs to standard out a cubic spline interpolation of npt evenly spaced points. The input file must be two columns giving X and Y(X). The derivatives at the endpoints are assumed to be zero. To change this, change ypl and ypn in the SetSpline function in the aflow_pflow_funcs.cpp file.

```
aflow --sp | --std_prim | --standard_primitive | --sprim < POSCAR
```

Output POSCAR in a standard primitive lattice

(use --sc | --std_conv | --standard_conventional | --sconv to output POSCAR in a standard conventional lattice) (WS & SC Nov09)

REF: Setyawan Curtarolo, DOI: 10.1016/j.commatsci.2010.05.010

```
aflow --statdiel OUTCAR*
```

Extracts and diagonalizes the static dielectric constant tensor from the output of a DFPT run (see the aflow "DIELECTRIC_STATIC" tag). The input OUTCAR can be in one of two formats: plain text (regular VASP output), or compressed with the bzip2 utility (OUTCAR*.bz2). The routines attempt to symmetrize the tensor to within a tolerance (currently set to 1.0e-5) prior to diagonalization.

The output is a single row containing 6 numbers. The first three are the real-valued diagonal terms of the tensor eps(xx), eps(yy), epz(zz); while the second set of three contain any imaginary eigenvalues that would result from a failure of the ad-hoc symmetrization procedure.

Example corresponding to Al1As1_ICSD_67784:

```
input: aflow --statdiel OUTCAR.dielectric_static.bz2
```

```
output: 8.772122 8.772122 8.772122 0.000000 0.000000 0.000000
```

```
aflow --suffix=[directory,]"from2to"
```

Change the suffixes of VASP files. Easy conversion between AFLOW format and VASP format. (KESONG Dec. 2009)

Mnemonic: from2to with from/to = [n=none;r1=relax1;r2=relax2;r3=relax3;s=static;b=bands] or without abbreviation

Changes suffix of all VASPFILES from VASPFILE*"from" to VASPFILE*"to".

If a directory is specified (e.g. aflow --suffix=abc,n2s)

then it change abc/VASPFILE*"from" to abc/VASPFILE*"to"

Example: aflow --suffix=n2s (or --suffix=none2static)

Change VASPFILE* to VASPFILE*.static

Example: aflow --suffix=s2n (or --suffix=static2none)

Change VASPFILE*.static to VASPFILE*

```
aflow --sumpdos pdos.in PROOUT
```

Works for vasp.46x

This allows you to sum up projected DOS together for convenient plotting. It only works when you have run with the following INCAR file settings. LORBIT=1 or 2 and set RWIGS, or LORBIT = 11 or 12 and no RWIGS (only works when using PAW PP).

If you are running parallel you must set NPAR=1. The input file is similar to --pdos above but somewhat simpler so I give a full example here.

```
# Input for aflow --sumpdos.
```

```
# These values you supply once.
```

```
SPIN = 1 # 1 = non-spin polarized, 2 = spin polarized
```

```

# Default 1
EFERMI = -999 # This will be subtracted from the energies.
# Set to 0 to subtract nothing, -999 to use
# the E_Fermi in the DOSCAR. Default -999.
NLM = 9 # number of orbitals, 9 (spd:1+3+5) or 16 (spdf:1+3+5+7)
# Default 9
PRINT_PARAMS = 0 # 0=prints only data (easy to plot).
# 1=prints all the input parameters.
# default: 0

# You can have as many cases as you want.
# They are all added together.

# case 1: t2g on atom 1
ATOMS = 1 # default: no atoms
LMVALUES = 5 6 8 # default: no lm
# case 2: eg on atoms 2 and 3
ATOMS = 2 3 # default: no atoms
LMVALUES = 7 9 # default: no lm

All # denote comment lines and can be put anywhere.
Each case is started when the token ATOMS is used.
Following an ATOMS token, all LMVALUES tokens
will apply to the atoms denoted in the preceeding
ATOMS token until the next ATOM token. You can have any number
of cases, and the results for all cases are added together.
WARNING: make sure to use SPIN to set if it is a spin polarized
calculation, make sure to use EFERMI to your desired reference,
and make sure to set NLM to 9 (spd) or 16 (spdf).
The lm values correspond to orbitals by the following scheme.
Input number: 1 2 3 4 5 6 7 8 9
Orbitals:      S Py Pz Px Dxy Dyx Dz2 Dxz Dx2-y2
Input number: 10 11 12 13 14 15 16
Orbitals:      F1 F2 F3 F4 F5 F6 F7
aflow --supercell=a11,a12,a13,a21,a22,a23,a31,a32,a33 < POSCAR
aflow --supercell=a11,a22,a33 < POSCAR
aflow --supercell=file < POSCAR
Outputs to standard out a supercell of the input POSCAR file.
This lattice vectors of the supercell are given by multiplying
the original cell parameters by the 3x3 matrix a_ij coefficients.
The supercell need not be integral combinations of the original
lattice vectors, although using fraction may cause you to end up
with a lattice inequivalent to your original. This can be used
to build big supercells, swap lattice vectors, etc.
The nine numbers must be separated spaces, and they form the
nine elements a11,a12,a13,a21,a22,a23,a31,a32,a33 of the 3x3
supercell matrix, respectively. If you specify only 3 numbers,
the other six are taken zero. If you use the "file" syntax,
nine numbers are read from file. They can be on one or
multiple lines. New algorithm by SC (aug07).
aflow --supercell_strlist=a11,a12,a13,a21,a22,a23,a31,a32,a33,strlist
aflow --supercell_strlist=a11,a22,a33,strlist
aflow --supercell_strlist=file,strlist
Outputs to standard out a sequence of structures in POSCAR format.
The structures are supercells formed using the supercell matrix
a11,a12,a13,a21,a22,a23,a31,a32,a33. If only 3 values are specified
the other are taken to be zero. If you use the "file" syntax,
nine numbers are read from file. They can be on one or multiple
lines. For more information on supercells see --supercell.
For more information on a strlist see --make_strlist.

aflow --superlattice=structure_type,n_min,n_max < POSCAR
--superlattice=VASP,structure_type,A,B < superlattice_name
Generate a superlattice of base structure POSCAR
structure_type : fcc/bcc/hcp

```

n_min, n_max : minimum and maximum numbers of base structures in the superlattice

`aflow --swap specie1 specie2 < POSCAR`
 Swap the specie number 1 with the specie number 2.
 It is useful if you forgot to put the POSCAR in alphabetic mode.

`aflow --terdata=Aa:Bb:Cc [--fonts=XX | --keep=eps | --print=jpg | --print=gif | --print=png]`
 Inputs are the elements in alphabetic order. Output is ternary convexhull named
 "phasediagram_AaBbCc.num.pdf". The num is the calculated number of compounds.
 EX:
`aflow --terdata=Nb:Pt:Rh`
 No matter the order of the elements, the list is alphabetized.
 Options
`--fonts=XX` specify the size of gnuplot fonts, default 50
`--keep=eps` avoid to delete the postscript before making the pdf
`--print=jpg` adds the jpg picture
`--print=gif` adds the gif picture
`--print=png` adds the png picture

`acovasp --terdata_exist list`
 The list is the one containing each element for the systems for a certain format.
 Each element should be alphabetical order. Put 2 strings before each elements.
 EX:
`aflow --terdata_exist system`

INPUTS EX:
`empty empty Al Co Mn`
`empty empty Co Ge Mn`

`aflow --terdata A B C list`
 Inputs are the elements in alphabetic order and extra data for the list. Output is ternary convexhull named
 "phasediagram_ABC.num.pdf". The num is the calculated number of compounds.
 list type is:
`ADDRESS1 COMPOUND_NAME1`
`ADDRESS2 COMPOUND_NAME2`
`... ..`
 EX:
`aflow --terdata terdata Nb Pt Rh list`

`aflow --uffenergy | --ue < POSCAR`
 Output the energy of the structure using the simplified Universal Force Filed (UFF) method [JACS, 114,

`aflow --vasp`
 Transforms the GEOMETRICAL (or whichever is the input file format) to a
 VASP format.

`aflow --volume v < POSCAR`
 Outputs POSCAR file giving having volume equal to v.

`aflow --volume*= x < POSCAR`
`aflow --volume+= x < POSCAR`
 Outputs POSCAR with volume changes as *=, += actor (like in c,c++).

`aflow --wyccar[=tolerance | =tight | =loose] < POSCAR`
 Prints the wyccar file (the poscar file with wyckoff positions) for the input POSCAR
 The wyccar format presents the lattice in standard conventional form and the wyckoff
 positions with their associated labels and site symmetry. If variability in the
 Wyckoff positions exists, the algorithm finds the Wyckoff positions with the "smallest"
 lettering scheme ("a" being the smallest). (R. Taylor/ D. Hicks)

`aflow --xray=lambda < POSCAR`
 Outputs to standard out the powder xray scattering pattern for
 the structure specified in the POSCAR input file. The
 wavelength of the scattering radiation is taken to be 1.
 The xray calculation is almost nothing more than the structure
 factor, although it includes approximate treatments of the

Debye-Waller terms and the Lorentz-polarization. In the xray calculations there are a number of terms that require a lot of information to really do right so I approximate them. First, scattering factors default to the atomic number. In some cases I coded more accurate values taken from tables for lambda=1.5418. No lambda dependence of the scattering factors is presently included. For the list of all coded atomic scattering factors see the constructor for the structure class in structure.cc. I include the Lorentz-polarization and an approximate Debye-Waller factor (DWF). The DWF is found assuming T=300, T_Debye=300, and within the high-temperature Debye approximation. The atomic mass defaults to twice the atomic number but I have input a few more accurate values in the code. For the list of all coded atomic masses see the constructor for the structure class in structure.cc. This simulation should give very accurate peak locations and qualitative relative integrated intensities (peak heights). I am still not happy with why the peak intensities do not reproduce other codes better. The output contains formats with and without peaks at the same 2theta grouped together. It also contains data ready for plotting. Each type of data has a keyword on every line so you can grep it out easily.

```
aflow --xrd_dist=h,k,l < POSCAR
  Compute Miller plane distance (distance between planes of atoms). The triplet h,k,l
  defines the plane spanned by the three points a/h,b/k,c/l, where a b c
  are the CONVENTIONAL lattice vectors. You may enter h,k,l as integers
  (e.g., 1 2 3), decimal numbers (e.g., 0.56 .4 .5), or fractions (e.g., 1/2 3/4 4/5)
  or any combination of the three. The POSCAR must be in conventional form to get
  meaningful results. You may first use the aflow command "aflow --sc < POSCAR" to get
  standard conventional. For example,
  cat POSCAR | aflow --sc | aflow --xrd_dist=1,0,0
aflow --xyz=[n1[,n2[,n3]]] < POSCAR
  Outputs to standard out an xyz file based on the POSCAR input
  file. This can be used as input for rasmol, xmol, etc..
  If you want atom names you must put them after each atom
  position in the POSCAR file (see -names). If any names are
  missing they are defaulted to H.
aflow --xyzwignerseitz [--xyzws] < POSCAR
  Performs "aflow --xyz=1,1,1" but moves the images
  of the atoms in the Wigner-Seitz cell. This parameter
  is usefull to fight against VASP trend of moving atoms
  in different (but traslationally equivalent) positions
  of the unit cell. (SC 10Jan04).

aflow --zval[=directory]
  Outputs from the available POTCAR/OUTCAR the sum of ZVAL.
aflow --zval_atom[=directory]
  Outputs from the available POTCAR/OUTCAR and POSCAR/CONTCAR the ZVAL of the cell per atom.
aflow --zval_cell[=directory]
  Outputs from the available POTCAR/OUTCAR and POSCAR/CONTCAR the ZVAL of the whole cell.
```

FROZSL MODE

```
aflow --frozsl_vaspsetup_aflow | --frozsl_vaspsetup < FROZSL.output
  Takes the FROZSL.output (the one containing all representations
  and deformations) and makes the POSCAR part of the aflow.in.
  All the calculations are ready to be ran as they are enclosed
  in START.XXXX/STOP.XXXX frameworks and a AFLOW_POSTSCRIPT is
  generated to extract all the energies for the various directories.
  If another parameter is passed: i.e.
  aflow --frozsl_vaspsetup_aflow --file
  then the command used to make the POSCARs is printed.
  (Mike Mehl Dec 09).
aflow --frozsl_vaspsetup_poscar < FROZSL.output
```

Takes the FROZSL.output (the one containing all representations and deformations) and makes all the POSCARs.

If another parameter is passed: i.e.

aflow --frozsl_vaspsetup_poscar --file
then the command used to make the POSCARs is printed.

(Mike Mehl Dec 09).

aflow --frozsl_analyze < aflow.frozsl.out

aflow --frozsl_readme | --frozsl_help

A short introduction to frozsl calculations.

APENNSY MODE (this routines are available only in duke.edu domine)

aflow --xfixX system structure

Repairs the convex-hull calculation by removing
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB2))/LIB/RAW/system/structure
and copying
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB2))/LIB/LIB/system/structure
into

\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB2))/LIB/FIX/system/structure

You have to go by hand and check the problem (rare) and
probably clean and rerun the structure.

The exact command is included in pflow:XFIXX which
is inside aflow_pflow_main.cpp

ALL LIBRARIES MODES

aflow [--force] --lib2raw=all[,directory]

Searches all the subdirectories of 'pwd'/LIB/ and updates 'pwd'/RAW/ and 'pwd'/WEB/
entries for the missing ones. With directory specified, it searches on directory/LIB/
It works only on AFLOW_MATERIALS_SERVER (defined in aflow.h).
With "--force" it overwrites the old raw files if they exist.

LIB1/LIB2/LIB3/LIB4/LIB5/LIB6 MODE (this routines are available only in duke.edu domine)

Scripts for the production/maintenance of the LIB* libraries.

The "all" works only if 'pwd' = \$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB*))

aflow [--force] --lib2raw=directory

aflow [--force] --lib2raw=Al_h/bcc

From the \$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB*))/LIB/directory_including_lattice
calculation generates the RAW library
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_LIB*))/RAW/directory_including_lattice
which includes the files necessary for the LIB* project
analysis: aflow.in LOCK

If *static files are present, the DOS is calculated.

If *bands files are present, the electronic structure is calculaterd.

It works only on AFLOW_MATERIALS_SERVER (defined in aflow.h),

and it has been written for the LIB* project.

It can be used also for LIB* directories.

With "--force" it overwrites the old raw files if they exist.

ICSD LIBRARY MODE

Scripts for the production/maintenance of the ICSD library.

The "all" works only if 'pwd' = \$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD))

aflow [--force] --lib2raw=directory

aflow [--force] --lib2raw=FCC/La1Se1_ICSD_27104

From the \$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD))/LIB/directory_including_lattice
calculation generates the RAW library
\$(vAFLOW_PROJECTS_DIRS.at(XHOST_LIBRARY_ICSD))/RAW/directory_including_lattice
which includes the files necessary for the ICSD project
analysis: aflow.in LOCK DOSCAR.static. EIGENVAL.bands
KPOINTS.bands POSCAR.bands.

It works only on AFLOW_MATERIALS_SERVER (defined in aflow.h),
and it has been written for the ICSD project.

With "--force" it overwrites the old raw files if they exist.

