

AFLOW V 2998.05

```
*****
*
*           AFlow - STEFANO CURTAROLO MIT/DUKE 2003-2009
*           High Throughput Ab-initio Computing Project
*
*****
Contributors: 2000-2009, Stefano Curtarolo (aflow aconvasp apennsy)
              2002-2004, Dane Morgan (convasp)
              2007, Anton van der Ven (symmetry methods)
              2007-2009, Wahyu Setyawan (--rsm --edos --kband, --icsd*)
              2008, Roman Chepulskyy (--edos --kband)
              2008, Gus Hart (lattice reductions, prototypes)
              2009, Ohad Levy and Raymundo Arroyave (prototypes)
*****
LATEST VERSION OF THE FILE:      http://materials.duke.edu/auro/aconvasp.pdf
*****
```

aflow: aconvasp mode (Stefano Curtarolo)

```
aconvasp --help [-h]
    Gives this help information.
```

```
aconvasp --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.
```

```
aconvasp --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.
```

```
aconvasp --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 !
```

```
aconvasp --aflowin_proto_binary label specieA specieB volumeA volumeB potential_type
    --aflowin_proto_binary can be substituted with --aflow_proto or --aproto
    Creates automatic aflow.in.
    Generate the directory: ./AFLOWDATA/specieAspecieB/label/
    and create an aflow.in inside with the "label" structure from
    DMQC-HTQC or the GUS library project (depending on the 1st letter)
    with the species specieA and specieB and the volumes per atoms,
    "volumeA" and "volumeB" (real in A3).
    The general structures are saved in the files aflow_xproto*.cpp.
    "potential_type": is one of the vasp types and must be specified
    "pot_LDA, pot_GGA, potpaw_LDA, potpaw_GGA, potpaw_PBE".
    The aflow.in file follows some sort of template defined inside
    the aflow_aconvasp_aflow.cpp file. It is easier to fix that file than
```

to include complicate options to this command.
Be careful to include the various "_pv,_sv, etc" in the species so that the POTCAR makes sense.

Note: you can use --aflowin_proto instead of --aflowin_proto_binary
HIGH-THROUGHPUT NOTE: label, specieA, specieB, volumeA, volumeB 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
volumeA1,volumeA2,volumeA3 volumes of specieAs, the number must be identical to the number of specieAs otherwise you get an error
volumeB1,volumeB2 volumes of specieBs, the number must be identical to the number of specieBs otherwise you get an error
potential_type is identical for all the combinations

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.
Beware that it might take a long long while.
Rent a netflix james bond movie and be patient.

aconvasp --aflowin_proto_binary label specieA specieB volume potential_type
Same as above but volume is taken as volume per atom
(not the volume of the cell!)
Note: you can use --aflowin_proto instead of --aflowin_proto_binary
Note: you can use --aflowin_proto instead of --aflowin_proto_binary
HIGH-THROUGHPUT NOTE: since a single volume definition does not define a set of calculations, uniquely, this option is not HIGH-THROUGHPUT.
If you specify multiple strings, only the 1st one will be operated.

aconvasp --aflowin_proto_binary label specieA specieB potential_type
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).
Note: you can use --aflowin_proto instead of --aflowin_proto_binary
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
potential_type is identical for all the combinations
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.
Beware that it might take a long long while.
Rent a netflix james bond movie and be patient.

aconvasp --aflowin_proto_binary_missing label specieA specieB volumeA volumeB potential_type
--aflowin_proto_binary_missing can be substituted with --aflow_proto_missing or --aproto_missing
aconvasp --aflowin_proto_binary_missing label specieA specieB volume potential_type
aconvasp --aflowin_proto_binary_missing label specieA specieB potential_type
Same as before but tries to find out if the prototype has already been generated.

aconvasp --aflowin_proto_missing | --aflow_proto_missing
Creates automatic aflow.in only for the missing structures of the database.
It checks on /common/GNDSTATE/LIBRARYG/LIB/ of materials.duke.edu

aconvasp --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 neighbors within d are used. MAX_NUM_ANGLE=21 at present and can be set in aconvasp_print.cpp

aconvasp --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.

aconvasp --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).

1. "TRI" unique
2. "MCL" unique
3. "MCLC" unique according to bilbao
4. "ORC" unique
5. choice a) "ORCC1" "ORCC_a<b"
choice b) "ORCC2" "ORCC_b<a"
6. choice a) "ORCF1" "ORCF_invb2+invc2<inva2" for $1/b^2 + 1/c^2 < 1/a^2$
choice b) "ORCF2" "ORCF_inva2<invb2+invc2" for $1/a^2 < 1/b^2 + 1/c^2$
7. "ORCI" unique
8. "TET" unique
9. choice a) "BCT1";// "BCT_c<a"
choice b) "BCT2";// "BCT_a<c"
10. "RHL" unique
11. "HEX" unique
12. "CUB" unique
13. "FCC" unique
14. "BCC" unique

aconvasp [--np NP] --cages < POSCAR

aconvasp [--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 including 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 labels 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 environment.

aconvasp --cart [-c] < POSCAR

Outputs to standard out a new POSCAR file with atom positions in cartesian coordinates.

aconvasp --chgdiff CHGCAR1 CHGCAR2

This takes the difference between two CHGCAR files, outputting CHGCAR1-CHGCAR2. The output is in the form of a vasp445 CHGCAR. The output file is CHGCAR.diff.out.

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).
aconvasp --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.
aconvasp --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.
aconvasp --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.
aconvasp --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 neighbors of each
  pair type out to rcut, differences in bond lengths for neighbors
  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).
aconvasp --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:
  aconvasp --compare ' aconvasp --ndata < POS1' ' aconvasp --ndata < POS2'

aconvasp --conventional | --conv < POSCAR
  Put the structure in conventional lattice (make it bigger if necessary).
aconvasp --conventional_primitive | --cprim
  Put the structure in the conventional primitive lattice so the reciprocal
  vectors define well the high-simetry points of the Brillouin Zone.

aconvasp --data < 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.
aconvasp --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.
aconvasp --data2 < POSCAR
  Outputs to standard out basic data about the structure in the
  POSCAR input file. Similar to --data but with another style (SC).
aconvasp --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
  neighbor is in.

```

`aconvasp --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 neighbor is in.

`aconvasp --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`

`aconvasp --edos ispin s d`
`aconvasp --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

`aconvasp --equivalent | --equiv | --iatoms < POSCAR`
 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).

`aconvasp --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 0 0 +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 `aconvasp` 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.

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

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

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

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

`aconvasp --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).

`aconvasp --factorgroup < POSCAR`
 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.

`aconvasp --findsym [tolerance_relative: default 1.0e-5] < POSCAR`
`aconvasp --frac [-f,-d,--frac,--fractional,--direct] < POSCAR`
 Outputs to standard out a new POSCAR file with atom positions in direct (fractional) coordinates.

`aconvasp --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.

aconvasp --hkl h k l < POSCAR

aconvasp --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).

aconvasp --hkl_search khlmax < POSCAR

aconvasp --hkl_search khlmax bond < POSCAR

aconvasp --hkl_search khlmax bond step < POSCAR

aconvasp --hkl_search < 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 hklmax, then the search is

-hklmax<=h<=hklmax, -hklmax<=k<=hklmax, -hklmax<=l<=hklmax

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.

aconvasp --hkl_search_simple < POSCAR

aconvasp --hkl_search_simple cutoff < POSCAR

aconvasp --hkl_search_simple cutoff bond < POSCAR

aconvasp --hkl_search_simple cutoff bond khlmax < POSCAR

aconvasp --hkl_search_simple cutoff bond khlmax step < POSCAR

aconvasp --hkl_search_complete cutoff < POSCAR

aconvasp --hkl_search_complete cutoff < POSCAR

aconvasp --hkl_search_complete cutoff bond < POSCAR

aconvasp --hkl_search_complete cutoff bond khlmax < POSCAR

aconvasp --hkl_search_complete cutoff bond khlmax 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 v1,v2,v3

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 cutoff³ !

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).
 HKLMAX (default "dims" given by cutoff)
 If you specify a hklmax, then the search is -hklmax<=h<=hklmax, -hklmax<=k<=hklmax, -hklmax<=l<=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).

aconvasp --icsd Pb Sn Se < ternary.icsd
 aconvasp --icsd Sn 34 Pb < ternary.icsd
 aconvasp --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)

aconvasp --icsd_alllessthan Ra < input.icsd
 (see --icsd). Output compounds if ALL elements in a compound have Z<Z_Ra

aconvasp --icsd_allmorethan Ra < input.icsd
 (see --icsd_alllessthan).

aconvasp --icsd_basislessthan 20 < input.icsd
 aconvasp --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.

aconvasp --icsd_chem MgB4 < input.icsd
 (see icsd). Extract MgB4 compound.

aconvasp --icsd_cubic < ternary.icsd
 aconvasp --icsd_triclinic < ternary.icsd
 aconvasp --icsd_monoclinic < ternary.icsd
 aconvasp --icsd_orthorhombic < ternary.icsd
 aconvasp --icsd_tetragonal < ternary.icsd
 aconvasp --icsd_rhombohedral < ternary.icsd
 aconvasp --icsd_trigonal < ternary.icsd
 aconvasp --icsd_hexagonal < ternary.icsd
 aconvasp --icsd_cubic --icsd_orthorhombic < ternary.icsd
 (see --icsd). Output all compounds which belong to the specified system.
 If multiple options, the relational is "OR".

aconvasp --icsd_tri < input.icsd
 aconvasp --icsd_mcl < input.icsd
 aconvasp --icsd_mclc < input.icsd
 aconvasp --icsd_orc < input.icsd
 aconvasp --icsd_orcc < input.icsd
 aconvasp --icsd_orcf < input.icsd
 aconvasp --icsd_orci < input.icsd
 aconvasp --icsd_tet < input.icsd
 aconvasp --icsd_bct < input.icsd
 aconvasp --icsd_rhl < input.icsd
 aconvasp --icsd_hex < input.icsd
 aconvasp --icsd_cub < input.icsd
 aconvasp --icsd_fcc < input.icsd
 aconvasp --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)

aconvasp --icsd_denslessthan 4.5 < ternary.icsd

aconvasp --icsd_densmorethan 5.5 < ternary.icsd

aconvasp --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.

aconvasp --icsd_id 11120 < input.icsd
 (see --icsd). Extract compound with entry ID number 11120 of the original ICSD database.

aconvasp --icsd_lessthan Pb < binary.icsd

aconvasp --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).

aconvasp --icsd_listmetals
 Output to stdout a list of metallic elements used in the --icsd_removemetals option.

aconvasp --icsd_morethan Pb < binary.icsd

aconvasp --icsd_morethan 82 < binary.icsd

aconvasp --icsd_morethan Pt --icsd_lessthan Hg < binary.icsd

aconvasp --icsd_morethan 78 --icsd_lessthan 80 < binary.icsd
 (see --icsd_lessthan and --icsd).

aconvasp --icsd_n_ary 2 < input.icsd
 (see --icsd). Output all binary compounds

aconvasp --icsd_n_ary 3 < input.icsd
 (see --icsd). Output all ternary compounds. and so on

aconvasp --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.

aconvasp --icsd_nopartialocc < input.icsd
 (see --icsd). Output only compounds with full occupancies

aconvasp --icsd_proto #Nspecies1 #Nspecies2 #Nspecies3 ... < input.icsd

aconvasp --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.

aconvasp --icsd_remove Pu < input.icsd
 (see --icsd). Remove compounds which contain Pu.

aconvasp --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

aconvasp --icsd_sg 62 < binary.icsd

aconvasp --icsd_sglessthan 62 < binary.icsd

aconvasp --icsd_sgmorethan 45 < binary.icsd

aconvasp --icsd_sgmorethan 194 --icsd_sglessthan 231 < binary.icsd
 (see --icsd). Output all compounds where the space group number satisfies the specified arguments.

aconvasp --icsd_unique < input.icsd
 (see --icsd). Dicard redundant compounds based on SG# AND ChemicalFormula

aconvasp --icsd2aflowin < input.icsd
 Convert input.icsd (ICSD-format) to aflow.in

aconvasp --icsd2poscar < input.icsd

Write output to stdout POSCAR from stdin ICSD-format
aconvasp --icsd2proto < input.icsd

Write output to stdout in ICSD_PROTO-format from stdin ICSD-format
aconvasp --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.

aconvasp --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.

aconvasp --icsdproto2aflowin < input.proto
Convert ICSD-PROTOTYPE-format to aflow.in
(2007-2009, wahyu@alumni.duke.edu)

aconvasp --identical < POSCAR
Makes all the atoms identical. Useful to find the properties of
the superlattice.

aconvasp --incell < POSCAR
Outputs to standard out a POSCAR file with all
atoms mapped to their images within the unit cell.

aconvasp --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)

aconvasp --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)

aconvasp --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.

aconvasp --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)

aconvasp --kpoints KDENS [--kppra,-k] < 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.

`aconvasp --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`.

`aconvasp --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.

`aconvasp --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 search, as it has been included by default inside. (SC0902).

`aconvasp --ltcell a11 a12 a13 a21 a22 a23 a31 a32 a33 < POSCAR`
`aconvasp --ltcell a11 a22 a33 < POSCAR`
`aconvasp --ltcell file < POSCAR`
 Outputs to standard out the linear transform of the input POSCAR file. This simply 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).

`aconvasp --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.

`aconvasp --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.

`aconvasp --minkowski_basis_reduction | --minkowski < 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>

`aconvasp --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.

`aconvasp --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.

`aconvasp --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.

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

`aconvasp --nanoparticle radius distance < POSCAR`
 Starting from the input poscar, aconvasp 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 the middle of the original unit cell. 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)

`aconvasp --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 V^{1/3} and phi are in degrees.

`aconvasp --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!).

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

`aconvasp --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.

```

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

```

aconvasp --phonons < POSCAR
aconvasp --phonons radius < POSCAR
Working. If radius is not specified, than the default is taken..
SC Dec07.

```

```

aconvasp --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).

```

```

aconvasp --platon [EQUAL] [EXACT] [ang d1 d2 d3] < POSCAR | platonSG
This finds the space group. aconvasp 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 0.45, 0.25 Angstrom).

```

d3 - Distance criterium for coinciding atoms for (pseudo) translation symmetry (default 0.45, 0.25 Angstrom).

To get the space group, type
`aconvasp -platon < POSCAR | platonSG`

To just get see the output file from aconvasp, type
`aconvasp -platon < POSCAR`

To check for errors and see output from platon, type
`aconvasp -platon < POSCAR | platon -o`

Note that the added flags above do not seem to work. To change tolerance create output file from aconvasp, and then add the four tolerances after CALC ADDSYM (on the same line). Then pipe this file to platonSG. Aconvasp 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 xdrv.c > 51108
 platon must be accessible on the path !

`aconvasp --platonSG [EQUAL] [EXACT] [ang d1 d2 d3] < POSCAR`
 This prints out the space group without bothering with the platonSG wrap up.
 Note: it works with platon.f and xdrv.c > 51108
 platon must be accessible on the path !

`aconvasp --platonSG_label [EQUAL] [EXACT] [ang d1 d2 d3] < POSCAR`
 Same as above but prints ONLY the space group name.

`aconvasp --platonSG_number [EQUAL] [EXACT] [ang d1 d2 d3] < POSCAR`
 Same as above but prints ONLY the space group number.

`aconvasp --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 aconvasp 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 aconvasp (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 aconvasp. I am not sure that this will all work the same way with PAW potentials.

```
aconvasp --pointgroup < POSCAR
```

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

```
aconvasp --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.

aconvasp --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 convasp appraoce).

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

aconvasp --proto_binary label specieA specieB volumeA volumeB
  Returns the structure labeled with the string label, from the
  DMQC-HTQC or the GUS library project (depending on the 1st letter)
  The atoms are called "specieA" and "specieB" (strings).
  The volumes per atoms are "volumeA" and "volumeB" (real in A^3).
  The general structures are saved in the file afLOW_xproto.cpp.
  Note: you can use --proto instead of --proto_binary
aconvasp --proto_binary label specieA specieB volume
  Same as above but volume is taken as volume per atom
  (not the volume of the cell!)
  Note: you can use --proto instead of --proto_binary
aconvasp --proto_binary label specieA specieB
  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).
  Note: you can use --proto instead of --proto_binary
aconvasp --proto_binary label
  Same as above. Atoms as "A" and "B" and volumes are 1.0
  Note: you can use --proto instead of --proto_binary

aconvasp --proto_htqc_pure label specieA volumeA
  Returns the structure labeled with the string label, from the
  DMQC-HTQC project.
  The atom is called "specieA" (strings).
  The volume per atoms is "volumeA" (real in A^3).
  The general structures are saved in the file afLOW_xproto.cpp.
aconvasp --proto_htqc_pure label specieA
  Same as above but the volume per atoms are extracted from the
  most dense paw_pbe values (usually the ground state).
aconvasp --proto_htqc_pure label
  Same as above. Atom as "A" and volume is 1.0

aconvasp --proto_gus_pure label specieA volumeA
  Returns the structure labeled with the string label, from the
  Gus Hart numbering project.
  The atom is called "specieA" (strings).
  The volume per atoms is "volumeA" (real in A^3).
  The general structures are saved in the file afLOW_xproto.cpp.

```

```

aconvasp --proto_gus_pure label specieA
    Same as above but the volume per atoms are extracted from the
    most dense paw_pbe values (usually the ground state).
aconvasp --proto_gus_pure label
    Same as above. Atom as "A" and volume is 1.0
aconvasp --proto_gus_cpp
    Create the "cpp" code for the structure loading in
    aflow_apennsy_gndstate.cpp (EXPERT SC0901)

aconvasp --prototypes_icsd | --protos_icsd
    Give the list of prototypes from the icsd database included in the code.
aconvasp --proto_icsd label
    Returns the prototype labeled with the string label, from the
    ICSD database.

aconvasp --rasmol n1 n2 n3 < POSCAR
aconvasp --rasmol < 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, aconvasp takes "1 1 1". (SC/OCT07)
aconvasp --raytrace rtfilerfile
    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 rtfilerfile
    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 ouput 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 # atomstshade 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 aconvasp and the user has not control over it from the rtfile. 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").

aconvasp --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. This 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_aconvasp_funcs.cpp.

`aconvasp --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.

`aconvasp --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 neighbor, including a sum of all neighbor 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 the increase sigma until you group desired atoms together. sigma=0 does no smearing.

`aconvasp --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 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.

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

aconvasp --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.

aconvasp --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)

aconvasp --rsm --z z_type1 z_type2 ... z_typeNtypes < POSCAR
Similar to aconvasp -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)

aconvasp --sc < POSCAR
or --std_conv or --standard_conventional
Output POSCAR in a standard conventional lattice
(use --sp or --std_prim or --standard_primitive to output POSCAR in a standard primitive lattice) (WS & SC Nov09)

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

aconvasp --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)

aconvasp --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.

aconvasp --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.

aconvasp --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

```

aconvasp --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, I
reduce 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!).
aconvasp --shift Sx Sy Sz [cCdD] < POSCAR
Outputs to standard out a POSCAR with all positions shifted by
S=(Sx,Sy,Sz). 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.
aconvasp --sitepointgroup < POSCAR
Calculates the site point group symmetry for every atom in the
unit cell and writes it in the aflow.agroup.out file.
See documentation of aflow.
aconvasp --spacegroup radius < POSCAR
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 radius^3 times the size of the factor roup.
The point and factopr groups are required for the space group,
therefore the aflow.pgroup.out and aflow.fgroup.out files will
be generated as well.

aconvasp --species < POSCAR
Outputs the species names, as taken from the 1st representative
atom of each set.
aconvasp --spline npt < file
Outputs to standard out a cubic spline interpolation of npt
evenly spaced points. The infput file must be two colums
giving X and Y(X). The derivatives at the endpoints are
assumed to be zero. To change this, change yp1 and ypn
in the SetSpline function in the aflow_aconvasp_funcs.cpp file.
aconvasp --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 aconvasp --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
aconvasp --supercell a11 a12 a13 a21 a22 a23 a31 a32 a33 < POSCAR
aconvasp --supercell a11 a22 a33 < POSCAR
aconvasp --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).
aconvasp --supercell_strlist a11 a12 a13 a21 a22 a23 a31 a32 a33 strlist
aconvasp --supercell_strlist a11 a22 a33 strlist
aconvasp --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.

aconvasp --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.
aconvasp --volume v < POSCAR
  Outputs POSCAR file giving having volume equal to v.
aconvasp --volume*= x < POSCAR
aconvasp --volume/= x < POSCAR
aconvasp --volume+= x < POSCAR
aconvasp --volume-= x < POSCAR
  Outputs POSCAR with volume changes as *=, /=, +=, and -=
  factor (like in c,c++).

```

aconvasp --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.

aconvasp --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.

aconvasp --xyzwignerseitz [--xyzws] < POSCAR

Performs "convasp -xyz 1 1 1" but moves the images of the atoms in the Wigner-Seitz cell. This parameter is useful to fight against VASP trend of moving atoms in different (but translationally equivalent) positions of the unit cell. (SC 10Jan04).

APENNSY MODE

aconvasp --xrawN

Creates the LIBRARYN/RAW from the LIBRARYN/LIB files.

aconvasp --xrawG

Creates the LIBRARYG/RAW from the LIBRARYG/LIB files.

aconvasp --xrawU

Creates the LIBRARYU/RAW from the LIBRARYU/LIB files.

aconvasp --xrawX

Creates the LIBRARYX/RAW from the LIBRARYX/LIB files.

aconvasp --xfixN system structure

Repairs the convex-hull calculation by removing
/common/GNDSTATE/LIBRARYN/LIB/RAW/system/structure
and copying
/common/GNDSTATE/LIBRARYN/LIB/LIB/system/structure
into
/common/GNDSTATE/LIBRARYN/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 AConvaspXFIXN which is inside aflow_aconvasp_main.cpp

aconvasp --xfixG system structure

Same as before but for LIBRARYG.

aconvasp --xfixU system structure

Same as before but for LIBRARYU.
aconvasp --xfixX system structure
Same as before but for LIBRARYX.

SCINT MODE

Scripts for the production/maintenance of the SCINT library.

aconvasp --xraw_scint_single directory_including_lattice:

aconvasp --xraw_scint_single FCC/La1Se1_ICSD_27104

From the /common/SCINT/ICSD/LIB/directory_including_lattice calculation generates the RAW library /common/SCINT/ICSD/RAW/directory_including_lattice which includes the files necessary for the SCINT project analisis: aflow.in LOCK DOSCAR.static. EIGENVAL.bands KPOINTS.bands POSCAR.bands.

It works only on nietzsche.mems.duke.edu (materials.duke.edu), and it has been written for the SCINT project.

aconvasp --xraw_scint_all | --xraw_scint

Searches all directories /common/SCINT/ICSD/LIB/ and updates the RAW library /common/SCINT/ICSD/RAW/ for the missing ones.

It works only on nietzsche.mems.duke.edu (materials.duke.edu), and it has been written for the SCINT project.

SCRIPTING MODE

aconvasp --justbefore "string_to_find" < something

Prints all the strings (defined by EOL) present in the file (also stdout) "something" until the first occurrence of "string_to_find".

aconvasp --justafter "string_to_find" < something

Prints all the strings (defined by EOL) present in the file (also stdout) "something" after the first occurrence of "string_to_find".

aconvasp --qdel aaa,nnn:mmm,aaa,bbb,ccc

qdel's all the jobs specified between "," and the list of jobs between ":" in increments of 1.

For instance aconvasp --qdel 10:20,99,102

qdel's 10,11,12...20,99,102

aconvasp --qsub N file

submits file N times (useful for rapid aflow submissions)
