

AFLOW V 3.1.111

```
*****
*
*           aflow - STEFANO CURTAROLO Duke University 2003-2017
*           High-Throughput ab-initio Computing Project
*
*****
LATEST VERSION OF THE FILE:           materials.duke.edu/AFLOW/aflow.pdf
*****
```

aflow

```
-h | --help | --readme_aflow      This help
-v | --version                    Version Information
--machine                         Machine Information

--DIRECTORY[=| ]dir | --D[=| ]dir | --d[=| ]dir
                                Run the directory and its subdirectories
--FILE[=| ]file | --F[=| ]file | --f[=| ]file
                                List of directories to run from file
                                containing a list of aflow.in one per line.
--quiet | -quiet | -q           Quiet: remove all '00000 MESSAGES'
-c | --clean                      cleans everything except aflow.in
                                (if .gz or .bz2 inputs are present, they are decompressed)

--run                             Run only this directory.

--run=multi                       Find and search all subdirectories and
                                and run all the aflow.in without LOCK

--run=N                           Find and search all subdirectories and
                                and run the first N available aflow.in without LOCK.
                                The old option --runone is mapped into --run=1.

--loop                            When finish, wait for more run
--sort | -sort                   Sorts the aflow.in in the list
--reverse | -rsort              Reverse the aflow.in in the list
--random | -rnd                 Randomize the aflow.in in the list
--force | -force                Run the aflow.in even if the entry
                                is already in the database.

--mem=XX | --maxmem=XX          If XX is specified (considered in %), then aflow
                                tries to kill all the vaspXX (and mpivaspXX) using
                                more than XX% of the available RAM.
                                The killing process is tried while printing "Messages"
                                and "Temperature monitoring". Useful for the jobs managing.

--use_aflow.in=XXX              Uses XXX instead of "aflow.in" in searching/running/operating directories.
                                The option is very useful for compounded calculations.

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

--readme=aflow                  AFLOW help
--readme=aconvasp              AFLOW_ACONVASP help
--readme=apennsy               AFLOW_APENNSY help
--readme=apl                   AFLOW_APL help
--readme=qha                   AFLOW_APL help
--readme=aapl                  AFLOW_APL help
--readme=frozsl                AFLOW_FROZSL help

--np=NUMBER With MPI=ON (option or supercomputer defaults)
            aflow uses NUMBER as parameter for the MPI run.
            This declaration overrides the [AFLOW_MODE_MPI_MODE]NCPUS=XX
            statement in the aflow.in file.
```

With MPI=OFF
 aflow starts a NUMBER number of concurrent threaded runs within
 the "loop" search (--DIRECTORY=..) or the "file" table list (--FILE=..).
 For the multithread start, you have to enable multi.
 Examples:
 aflow --multi --np=32 --DIRECTORY=./ICSD_POOL
 aflow --multi --np=24 --FILE=./jobs2run

With aconvasp.
 Some parts of aflow are multithreaded. If you specify
 --np=NUMBER, a NUMBER of independent threads will be
 launched to perform the task in a multitasking fashion.

--npmax With this option, aflow starts a number of concurrent runs
 equal to the maximum numbers of processors.

--multish Very useful option for scripting.
 if you have a "file" containing a gazillion of instructions,
 one per line, such as
 command_perform ./directory1
 command_perform ./directory2

 command_perform ./directory..
 then the instruction
 aflow --multish --np=XX --FILE file
 will pipe the instruction in a push-pop list and feed XX cpus in a
 multithreaded environment. You can substitute "--np=XX" with "--npmax"
 (or omit it at all) and use the max number of available cores in the
 machine. Instead of --FILE you can use " --F | -F | --f | -f".
 If you omit the --FILE option, then the last argument will
 be taken as the name of the file.
 Note that due to the variable nature of the time requested for each line,
 you might lose causality in the whole process (most of the times
 you do not need it, though).
 These are examples of proper commands:
 aflow --multish --np=12 --FILE file (run file in 12 cores)
 aflow --multish --np=12 file (run file in 12 cores)
 aflow --multish file (run file in all your cores)

--multizip Very useful option for scripting.
 aflow --multizip [--prefix=PREFIX] [--size=SSSS] [--add] --F file | -D directory1 directory2 ..
 if you have a huge amount of directories to zip then you can clusterize them, so that
 each subzip contains no more than SSSS entries.
 The "prefix=" is optional, the default is "m"
 The "size=" is optional, the default is 100.
 If you add --add then, if zips are present, they will be added with the new files.
 This is a very useful option because many file systems do not allow big files.
 I usually use:
 aflow --multizip --prefix=magnetic --size=500 'find . -name EIGENVAL.bands.bz2'
 To make life easier, this option clears up the words
 "LOCK,aflow.in,OUTCAR.relax2.bz2,EIGENVAL.bands.bz2" from the directory names.

--multibzip2 Very useful option for scripting.
 aflow --multibzip2 --np=XX --FILE file1 file2 file3...
 if you have a huge amount of files to bzip2 then you can multithread the bzip2 so there are XX
 If --np= is not specified then the code will take 1 core.
 The filenames with ".bz2" extension are neglected.

--multibunzip2 Very useful option for scripting.
 aflow --multibunzip2 --np=XX --FILE file1.bz2 file2.bz2 file3.bz2...
 Same as --multibzip2 but for un-bzipping.
 The filenames without ".bz2" extension are neglected.

--multigzip Very useful option for scripting.
 aflow --multigzip --np=XX --FILE file1 file2 file3...
 if you have a huge amount of files to gzip then you can multithread the gzip so there are XX (n

If --np= is not specified then the code will take 1 core.
The filenames with ".gz" extension are neglected.

--multibunzip Very useful option for scripting.
aflow --multibunzip --np=XX --FILE file.gz file2.gz file3.gz....
Same as --multigzip but for un-bzipping.
The filenames without ".gz" extension are neglected.

--getTEMP [--runstat | --runbar | --refresh=X | --warning_beep=T | --warning_halt=T | --mem=XX]
If available, the command outputs the hostname and temperatures of the machine. Useful to find
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 console
with --warning_halt=T, if the max temp goes beyond T(C, DEFAULT below), the command halts the console
with --maxmem=X | --maxmem=XX, it kills vasp/mpivasp using more than XX% of memory.
DEFAULT VALUES in aflow.h
#define AFLOW_CORE_TEMPERATURE_BEEP_ 56.0 // Celsius
#define AFLOW_CORE_TEMPERATURE_HALT_ 65.0 // Celsius, you need to run aconvasp as root to
#define AFLOW_CORE_TEMPERATURE_REFRESH_ 5.0 // seconds

--monitor [--mem=XX]
This is a wrap up set of commands to be sent in the background so that the node is monitored for
It kills vasp/mpivasp using more than XX% of memory.
The default for XX is 95%/NCPUs, so even in the worst scenario there should be enough RAM to resuscitate
a soon-to-be-frozen machine/node.

--generate_aflowin_from_vasp
ACTION: Generates aflow.in from xCARs
NOTE1: You can add extra parameters to aflow.in by using
--set "[KEYWORD]", where the keyword is one
of the specified below. You can add as many as you want.
Be careful: there is no check for inconsistency.
NOTE2: You can remove VASP files after the generation with the
option --delete_xcars (remove all except aflow.in).

--generate_vasp_from_aflowin | --generate
ACTION: Generates xCARs from aflow.in
NOTE: This option does not run any simulation.

//DX and CO - START
--generate_symmetry | --generate_sym
ACTION: Generates symmetry files: aflow.pgroup.out, aflow.fgroup.out, aflow.pgroup_xtal.out, aflow.iautor
NOTE: This option does not run any simulation.
//DX and CO - END

MPI/SERIAL PARAMETERS
--mpi Force turn ON MPI
--nompi | --serial Force turn OFF MPI

HOST ORIENTED OPTION
--machine=beta
With this option, aflow tunes the MPI commands to "duke_beta_mpich" standards
OPTIONS ="ulimit -s unlimited "
COMMAND ="/usr/bin/mpiexec -np" (with mpich2)
BINARY_DIRECTORY ="/usr/local/bin/"
These parameters can be changed in aflow.h

--machine=beta_openmpi
With this option, aflow tunes the OPENMPI commands to "duke_beta_openmpi" standards
OPTIONS ="ulimit -s unlimited "
COMMAND ="/usr/bin/mpirun.openmpi"
BINARY_DIRECTORY ="/usr/local/bin/"
These parameters can be changed in aflow.h

--machine=qrats

```

        With this option, aflow tunes the MPI commands to "duke_qrats_mpich" standards
        OPTIONS ="ulimit -s unlimited "
        COMMAND ="/usr/bin/mpiexec.gforker -np " (with mpich2)
        BINARY_DIRECTORY ="/MAIN/bin/VASP/"
        These parameters can be changed in aflow.h

//DX and CO - START
--machine=quser
    With this option, aflow tunes the MPI commands to "duke_quser_openmpi" standards
    OPTIONS ="ulimit -s unlimited "
    COMMAND ="/usr/bin/mpirun -n " (with openmpi)
    BINARY_DIRECTORY ="/home/bin/"
    These parameters can be changed in aflow.h
//DX and CO - END

--machine=materials
    With this option, aflow tunes the MPI commands to "duke_materials" standards
    OPTIONS ="ulimit -s unlimited "
    COMMAND ="/usr/bin/mpiexec -np" (with mpich2)
    BINARY_DIRECTORY ="/usr/local/bin/"
    These parameters can be changed in aflow.h

--machine=afloplib
    With this option, aflow tunes the MPI commands to "duke_afloplib" standards
    OPTIONS ="ulimit -s unlimited "
    COMMAND ="/usr/bin/mpiexec -np" (with mpich2)
    BINARY_DIRECTORY ="/usr/local/bin/"
    These parameters can be changed in aflow.h

--machine=habana
    With this option, aflow tunes the MPI commands to "duke_habana" standards
    OPTIONS ="ulimit -s unlimited "
    COMMAND ="/usr/bin/mpiexec -np" (with mpich2)
    BINARY_DIRECTORY ="/usr/local/bin/"
    These parameters can be changed in aflow.h

--machine=ranger
    With this option, aflow tunes the MPI commands to "teragrid_ranger" standards
    COMMAND ="/share/sge6.2/default/pe_scripts/ibrun"
    BINARY_DIRECTORY ="/share/home/00457/tg457357/bin/"
    These parameters can be changed in aflow.h

--machine=kraken
    With this option, aflow tunes the MPI commands to "teragrid_kraken" standards
    COMMAND ="aprun -n"
    BINARY_DIRECTORY ="/nics/a/proj/aflow/bin/"
    These parameters can be changed in aflow.h

--machine=marylou
    With this option, aflow tunes the MPI commands to "fulton_marylou" standards
    OPTIONS = "export OMP_NUM_THREADS=1"
    COMMAND = "mpiexec"
    BINARY_DIRECTORY ="/fslgroup/fslg_datamining/bin/"
    These parameters can be changed in aflow.h

--machine=parsons
    With this option, aflow tunes the MPI commands to "trinity_parsons" standards
    COMMAND ="mpirun -np "
    BINARY_DIRECTORY ="/home/users/aflow/bin/"
    These parameters can be changed in aflow.h

--machine=jellium
    With this option, aflow tunes the MPI commands to "nrl_jellium" standards
    COMMAND = .. none, unnecessary
    BINARY_DIRECTORY ="/share/apps/AFLOW2/bin/"

```

These parameters can be changed in aflow.h

```
--machine=raptor --np=N
    With this option, aflow tunes the MPI commands to "raptor" standards
    COMMAND = "aprun -n N"
    BINARY_DIRECTORY = "~/bin/"
    These parameters can be changed in aflow.h

--machine=diamond --np=N
    With this option, aflow tunes the MPI commands to "diamond" standards
    COMMAND = "aprun -n N"
    BINARY_DIRECTORY = "~/bin/"
    These parameters can be changed in aflow.h

--machine=ohad
    With this option, aflow tunes the MPI commands to "ohad" standards
    COMMAND = .. none, unnecessary
    BINARY_DIRECTORY = "/home/aflow/bin/"
    These parameters can be changed in aflow.h

--machine=host1
    With this option, aflow tunes the MPI commands to "host1" standards
    COMMAND = "???"
    BINARY_DIRECTORY = "??????"
    These parameters can be changed in aflow.h
```

SCRIPTING ORIENTED OPTIONS

```
--cv      Explain this
```

```
*****
aflow.in
```

Aflow/aflowd reads lines starting with "[xxxx.." where xxx is the command. If you put a character between "[" and "xxxx" like "[!xxxx" the line is ignored. If you add a "#" at the beginning of a line (such as "#[xxx"), everything on the left of # is ignored by aflow as command/string/parameter (ignored by aflow does not mean ignored by the binary code!). These features are useful if you want to generate a lot of similar aflow.in's and you want to add/remove options in a very short time.

[AFLOW] OPTIONAL

everything on the left contains comments

[AFLOW_MODE=****] MANDATORY

Different modes of AFLOW running.

```
[AFLOW_MODE]ALIEN is supported
[AFLOW_MODE=ALIEN] is supported
[AFLOW_MODE_ALIEN] is supported
[AFLOW_MODE]VASP is supported
[AFLOW_MODE=VASP] is supported
[AFLOW_MODE_VASP] is supported
[AFLOW_MODE]MATLAB is supported
[AFLOW_MODE=MATLAB] is supported
[AFLOW_MODE_MATLAB] is supported
[AFLOW_MODE]ENCAPSULATED is supported
[AFLOW_MODE=ENCAPSULATED] is supported
[AFLOW_MODE_ENCAPSULATED] is supported
```

[AFLOW_MODE_ZIP=****] or [AFLOW_MODE_ZIP]**** OPTIONAL, Default ****=gzip

Compression of output files at the end. You can write whatever you want as long as "command" is recognized.

I suggest:

```
[AFLOW_MODE_ZIP=none] does not compress
[AFLOW_MODE_ZIP=gzip]
```

[AFLOW_MODE_ZIP=bzip2]

[AFLOW_MODE_BINARY=****] or [AFLOW_MODE_BINARY]**** OPTIONAL, Default ****=vasp46s
the binary you want to start in each directory. If you put ./xxx
you would start the exact binary in such directory.
In MODE_VASP the default is vasp46s

[AFLOW_MODE_PRESCRIPT] OPTIONAL
Execute everything after the keywords [AFLOW_MODE_PRESCRIPT]
as a script BEFORE executing the AFLOW_MODE_BINARY simulations.
The output of the script is piped into the aflow.prescript.out file.
You need to watch for syntax and be careful with the commands.
The script can also be contained between
[AFLOW_MODE_PRESCRIPT]START
script
[AFLOW_MODE_PRESCRIPT]STOP

[AFLOW_MODE_POSTSCRIPT] OPTIONAL
Execute everything after the keywords [AFLOW_MODE_POSTSCRIPT]
as a script AFTER executing the AFLOW_MODE_BINARY simulations.
The output of the script is piped into the aflow.postscript.out file.
You need to watch for syntax and be careful with the commands.
The script can also be contained between
[AFLOW_MODE_POSTSCRIPT]START
script
[AFLOW_MODE_POSTSCRIPT]STOP

[AFLOW_MODE_EMAIL] or [AFLOW_MODE]EMAIL OPTIONAL
Email the recipient address specified after the keyword
[AFLOW_MODE_EMAIL] the output of the LOCK file once the
prescript, simulations and postscripts are execute.
WARNING: This keyword is not implemented yet.

HOST: fix default things for the HOST

[AFLOW_HOST]BETA is supported
With the BETA option, AFLOW takes the default mpirun command:
MPI_COMMAND_BETA = "/usr/bin/mpiexec", takes the BIN home for the MPI code as
MPI_BINARY_DIR_BETA = "/usr/local/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]BETA_OPENMPI is supported
With the BETA_OPENMPI option, AFLOW takes the default mpirun command:
MPI_COMMAND_BETA_OPENMPI = "/usr/bin/mpirun.openmpi", takes the BIN home for the MPI code as
MPI_BINARY_DIR_BETA_OPENMPI = "/usr/local/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]QRATS is supported
With the QRATS option, AFLOW takes the default mpirun command:
MPI_COMMAND_QRATS = "/usr/bin/mpiexec.gforker -np ", takes the BIN home for the MPI code as
MPI_BINARY_DIR_QRATS = "/MAIN/bin/VASP/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

//DX and CO - START
[AFLOW_HOST]QUSER is supported
With the QUSER option, AFLOW takes the default open-mpi command:
MPI_COMMAND_QUSER = "/usr/bin/mpirun -n ", takes the BIN home for the MPI code as
MPI_BINARY_DIR_QUSER = "/home/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)
//DX and CO - END

[AFLOW_HOST]MATERIALS is supported
With the MATERIALS option, AFLOW takes the default mpirun command:

MPI_COMMAND_MATERIALS = "/usr/bin/mpiexec", takes the BIN home for the MPI code as
MPI_BINARY_DIR_MATERIALS = "/usr/local/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]AFLOWLIB is supported

With the AFLOWLIB option, AFLOW takes the default mpirun command:

MPI_COMMAND_DUKE_AFLOWLIB = "/usr/bin/mpiexec", takes the BIN home for the MPI code as
MPI_BINARY_DIR_DUKE_AFLOWLIB = "/usr/local/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]HABANA is supported

With the HABANA option, AFLOW takes the default mpirun command:

MPI_COMMAND_DUKE_HABANA = "/usr/bin/mpiexec", takes the BIN home for the MPI code as
MPI_BINARY_DIR_DUKE_HABANA = "/usr/local/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]RANGER is supported

With the RANGER option, AFLOW takes the default mpirun command:

MPI_COMMAND_TERAGRID_RANGER = "/share/sge6.2/default/pe_scripts/ibrun", takes the BIN home for the MPI code as
MPI_BINARY_DIR_TERAGRID_RANGER = "/share/home/00457/tg457357/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]KRAKEN is supported

With the KRAKEN option, AFLOW takes the default ibrun command:

MPI_COMMAND_TERAGRID_KRAKEN = "aprun", takes the BIN home for the MPI code as
MPI_BINARY_DIR_TERAGRID_KRAKEN = "/nics/a/proj/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]PARSONS is supported

With the PARSONS option, AFLOW takes the default mpirun command:

MPI_COMMAND_TRINITY_PARSONS = "mpirun -np ", takes the BIN home for the MPI code as
MPI_BINARY_DIR_TRINITY_PARSONS = "/home/users/aflow/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]MARYLOU is supported

With the MARYLOU option, AFLOW takes the default mpirun command:

MPI_COMMAND_FULTON_MARYLOU = "/apps/openmpi/1.6.3_intel-13.0.1/bin/mpiexec", takes the BIN home for the MPI code as
MPI_BINARY_DIR_FULTON_MARYLOU = "/fslgroup/fslg_datamining/bin/", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_HOST]HOST1 is supported

With the HOST1 option, AFLOW takes the default ibrun command

MPI_COMMAND_HOST1 = "????????", takes the BIN home for the MPI code as
MPI_BINARY_DIR_HOST1 = "?????", sets up the AUTOTUNE (start,stop are neglected)
and overrides the NCPUS value specified with the option --np=NUMBER (if specified)

[AFLOW_MODE_MPI] OPTIONAL, default NONE

Turns ON MPI parallel execution. Aflow will neglect the
serial AFLOW_MODE_BINARY=binary.

The computer should have already all the files to run
mpi executables, such as hosts/.rhost/ etcetera.

[AFLOW_MODE_MPI_MODE]KEYWORD OPTIONAL

Keyword to specify parameters. After [AFLOW_MODE_MPI_MODE] you
can put

NCPUS=NNN OPTIONAL, default NNN=4

With NCPUS=0 or NCPUS=MAX aflow will try to guess the
maximum number of cpus of the system by performing and
analyzing the file /proc/cpuinfo as described in the note.

With NNN=1 aflow will revert to SERIAL execution with the
AFLOW_MODE_BINARY=binary.

Note: NCPUS=MAX goes through execution of the logical
command 'cat /proc/cpuinfo | grep -c "cpu MHz"'. It works
on Linux installations with 2.6+ kernels. In case of troubles

the default is `_MPI_NCPUS_DEF=4` specified in `aflow.h`.
This is good for old alphas (I have a beautiful Alpha ES45 with 4 CPUS. Stefano).

`START="string" OPTIONAL, default ""`

The command you have to perform to start the mpi daemon.
For LAM you have to execute "lamboot". For mpich1 or mpich2 you have to specify something else.

`STOP="string" OPTIONAL, default ""`

The command you have to perform to stop the mpi daemon.
For LAM you have to execute "lamhalt". For mpich1 or mpich2 you have to specify something else.

`COMMAND="string" OPTIONAL, default "mpirun -np"`

The command you have to perform to execute the mpi executable. Usually it is "mpirun" but in some computers with multiple installations you might need to specify something else.

`AUTOTUNE OPTIONAL`

If set, then aflow will neglect the PARALLEL MPI parameters in the input file and adjust them following the instructions of the code you are using.

Note: this flag is currently used only for VASP and tunes LPLANE,NPAR,IALGO,LSCALU,NSIM as specified in the VASP manual for Linux Clusters.

`BINARY="string" OPTIONAL, default "mpivasp46s"`

Specifies the mpi executable you are trying to run.

`[AFLOW_MODE_QSUB] OPTIONAL, default NONE`

Turns ON qsub (or other queue) systems. Aflow will go inside the directory and produce a file, "aflow.qsub.run", as described below.

The file MUST organize the queue batch script and, at the end, should contain 'echo "DONE" > aflow.qsub.done'.
Once the queue is submitted, the file "aflow.qsub.done" is checked every few minutes, until the string DONE is found and the simulation is considered finished. Check out the example below.

`[AFLOW_QSUB_MODE_**PLICIT] MANDATORY, no default`

Only the EXPLICIT mode of QSUB is supported. Everything on the left of "[AFLOW_QSUB_FILE]" strings is copied into QSUB files.

NOTE to [AFLOW_QSUB_MODE_EXPLICIT]

With [AFLOW_QSUB_MODE_EXPLICIT] activated, you can add:

[AFLOW_QSUB_MODE_EXPLICIT]START and

[AFLOW_QSUB_MODE_EXPLICIT]STOP

This helps because instead of specifying QSUB with the "[AFLOW_QSUB_FILE]" strings, everything between the [...]START and [...]STOP keys are copied inside a QSUB.

This option is very useful trick to cut/paste long QSUBs without adding the "[AFLOW_QSUB_FILE]" strings at the beginning of each line.

`[AFLOW_QSUB_MODE]COMMAND="string" OPTIONAL, default "qsub"`

This is the command that is used to submit a job
The default is qsub but you can change it for your needs.

`[AFLOW_QSUB_MODE]PARAMS="string" OPTIONAL, default nothing`

These are the parameters that are used to submit a job in the queue.

NOTE: For MPI, the queue works in the same way, but the NCPUS=MAX should be avoided because aflow can not predict in which "node" the job will run (you might get as NCPUS the number of CPUS of the master node !!). My best suggestion is that you put NCPUS as the number you want.

The MPI keywords, START/STOP/COMMAND/BINARY are neglected.

You should prepare the "aflow.qsub.run" by yourself inside aflow.in as the example below shows.
The MPI AUTOTUNE is performed, and this is very useful so the INCAR is automatically adapted for MPI jobs.

NOTE2: I wrote some shortcuts for common used batch systems.
With this parameters in aflow.in OR as command arguments, you can avoid writing long and tedious batch scripts inside the aflow.in

```
[AFLOW_QSUB_MODE]MODE1 equal to aflow --gsub1
[AFLOW_QSUB_MODE]MODE2 equal to aflow --gsub2
[AFLOW_QSUB_MODE]MODE3 equal to aflow --gsub3
(Model is for Gus-Stefano Mg project in Marylou4)
```

```
*****
[AFLOW_MODE=ALIEN] or [AFLOW_MODE_ALIEN] or [AFLOW_MODE]ALIEN
```

In MODE_ALIEN *****
You do not need to create ALIEN code files by hand. You can put everything inside aflow.in

```
[ALIEN_COMMAND]prog > output
Contains the command to run, it can contain the program and the output.
The input file can be contained in the EXPLICIT/IMPLICIT/EXTERNAL
For long commands you can make a script with
[ALIEN_COMMAND]START
prog1_do_something
prog2_do_something_else
[ALIEN_COMMAND]STOP
and all the output will be sent to "output". The START/STOP command
overrides the simple "[ALIEN_COMMAND]prog > output" specification.
```

```
[ALIEN_INPUT_FILE_**PLICIT]
Only the EXPLICIT mode of ALIEN is supported. Everything on
the left of "[ALIEN_INPUT_FILE]" strings is copied into an "INPUT" file
specified by the "INPUT=" keyword.
NOTE to [ALIEN_INPUT_FILE_EXPLICIT]
With [ALIEN_INPUT_FILE_EXPLICIT] activated, you can add:
[ALIEN_INPUT_FILE_EXPLICIT]START and
[ALIEN_INPUT_FILE_EXPLICIT]STOP
This helps because instead of specifying INPUT file with the
"[ALIEN_INPUT_FILE]" strings, everything between the [...]START
and [...]STOP keys are copied inside the INPUT file.
This option is very useful trick to cut/paste long INPUT files
without adding the "[ALIEN_FILE]"
strings at the beginning of each line.
```

```
[ALIEN_INPUT_FILE_EXTERNAL]
Searches a file or loads a file from stdout command:
[ALIEN_INPUT_FILE]FILE=../../somewhere/input
or
[ALIEN_INPUT_FILE]COMMAND=bzcat ../somewhere/input.bz2
if FILE or COMMAND are not specified, aflow takes the standard
FILE=./input as default.
```

```
[ALIEN_INPUT_FILE_NAME]INPUT=
Specifies the name of the input file for the general ALIEN program.
If not specified, aflow takes "./input" as default
```

```
[ALIEN_OUTPUT_FILE_NAME]OUTPUT=
Specifies the name of the output file for the general ALIEN program.
If not specified, aflow takes "./output" as default
```

```
*****
[AFLOW_MODE=MATLAB] or [AFLOW_MODE_MATLAB] or [AFLOW_MODE]MATLAB
```

In MODE_MATLAB *****

You do not need to create MATLAB code files
by hand. You can put everything inside aflow.in

[AFLOW_MATLAB_MODE_**PLICIT] MANDATORY, no default
Only the EXPLICIT mode of MATLAB is supported. Everything on
the left of "[AFLOW_MATLAB_FILE]" strings is copied into an "aflow.m" file.
This file is executed as "matlab -r aflow > aflow.out"
(check the binary location in the aflow.h file) and
the output is put in an "aflow.out" file.
This is usefull to create structures with matlab.
NOTE to [AFLOW_MATLAB_MODE_EXPLICIT]
With [AFLOW_MATLAB_MODE_EXPLICIT] activated, you can add:
[AFLOW_MATLAB_MODE_EXPLICIT]START and
[AFLOW_MATLAB_MODE_EXPLICIT]STOP
This helps because instead of specifying MATLAB code with the
"[AFLOW_MATLAB_FILE]" strings, everything between the ...]START
and ...]STOP keys are copied inside a MATLAB code.
This option is very useful trick to cut/paste long MATLAB
codes without adding the "[AFLOW_MATLAB_FILE]"
strings ad the beginning of each line.

[AFLOW_MATLAB_MODE_EXTERNAL]
Searches a file or loads a file from stdout command:
[AFLOW_MATLAB_FILE]FILE=../../somewhere/prog.m
or
[AFLOW_MATLAB_FILE]COMMAND=bzcat ../somewhere/prog.m.bz2
if FILE or COMMAND are not specified, aflow takes the standard
FILE=./prog.m as default.

[AFLOW_SYMMETRY] or [VASP_SYMMETRY]KEYWORD OPTIONAL
Keyword to specify parameters. After [AFLOW_SYMMETRY]
you can put
CALCULATION OPTIONAL
With this keyword aflow calculates:
* aflow.pgroup.out *: Point group of the lattice {R}
* aflow.fgroup.out *: Factor group of the cell {R|t},
note that this might not be a "true" group.
* aflow.agroup.out *: Site point group for every atomic
position (the maximum symmorphic subgrup of the lattice
point group applied to each atomic point).
* aflow.sgroup.out *: Space group {R|t+T} with T
up to a radius from the origin.
Files: the poing and factor groups are saved in
aflow.pgroup.outand aflow.fgroup.out files. The site
point group is saved in aflow.agroup.out.
The aflow.sgroup.out file is not saves unless the
following keyword is specified
SGROUP_WRITE OPTIONAL
Flag, if present, the space group is saved in a file
aflow.out.sgroup (use this with caution, the file might
get huge)
SGROUP_RADIUS=XXX OPTIONAL, default XXX=5.0
specifies the radius of a sphere around the origin
where the translations of the space group is calculated.
Be careful because the size of the space group increases
as the radius³ times the size of the factor roup.
//DX and CO - START
NO_SCAN OPTIONAL
The symmetry routine involves consistency checks. When
a symmetry rule is broken, the tolerance is changed and
recalculated. This keyword will not perform the scan if
symmetry rules are broken.

SYM_EPS OPTIONAL, default minimum_interatomic_distance/100.0
Specifies the tolerance for all symmetry routines.
(In Angstroms).
//DX and CO - END

[AFLOW_NEIGHBOURS]KEYWORD OPTIONAL
Keyword to specify parameters. After [AFLOW_NEIGHBOURS]
you can put
CALCULATION OPTIONAL
WRITE OPTIONAL
RADIUS=XXX OPTIONAL, default XXX=5.0
DRADIUS=XXX OPTIONAL, default XXX=0.1

[AFLOW_APL]CALC OPTIONAL
Calculate harmonic phonons. Read README_AFLOW_APL.TXT.
[AFLOW_QHA]CALC OPTIONAL
Calculate Gruneisen parameter via quasi-harmonic approximation. Read README_AFLOW_APL.TXT.
[AFLOW_AAPL]CALC OPTIONAL
Calculate anharmonic phonons. Read README_AFLOW_APL.TXT.

[AFLOW_MODE=VASP] or [AFLOW_MODE_MATLAB]

In MODE_VASP *****
You do not need to create INCAR/POSCAR/POTCAR/KPOINTS files
by hand. You can put everything inside aflow.in

[VASP_RUN]KEYWORDS .. where keywords can be

GENERATE OPTIONAL
Generate all the files, and no vasp is generated. All the XCARS are generated and tuned accordingly
(the original versions are backup in the XCARS.origs).

STATIC OPTIONAL
Performs a STATIC run (different than RELAX=0).
The keys IBRION,NSW,ISIF are commented in the INCAR file.

KPOINTS OPTIONAL
Runs a swap of kpoints from the small to the prescribed ones so that the calculations
relax faster (but run more relaxations). This option can be used for kpoints convergence.

RELAX=N OPTIONAL, Default N=2
Selects the number of relaxations RELAX=N with N=0 to 99999 (many relaxations!).
If N=0 then NO RUN is performed (XCARS are generated and twisted).

RELAX_STATIC=N OPTIONAL, Default N=2
Selects the number of relaxations RELAX_STATIC=N with N=0 to 99999 (many relaxations!).
If N=0 then NO RUN is performed (XCARS are generated and twisted).
After the N relaxations, a static run is performed with ad hoc INCARS.
Look for RELAX_STATIC options for tuning the calculations.

RELAX_STATIC_BANDS=N OPTIONAL, Default N=2
Selects the number of relaxations RELAX_STATIC_BANDS=N with N=0 to 99999 (many relaxations!).
if N=0 then NO RUN is performed (XCARS are generated and twisted).
After the N relaxations, a static run is performed with ad hoc
INCARS. After the STATIC run an BANDS calculation is performed.
Look for RELAX_STATIC_BANDS options for tuning the calculations.
You shall have a KPOINTS IMPLICIT and you must specify:
[VASP_KPOINTS_FILE]BANDS_LATTICE=fcc (cub,bcc,tet,bct,hex,orc,orci,orcc,orcf,rhl,tri,auto)
[VASP_KPOINTS_FILE]BANDS_GRID=16 (grid, thickness of vasp kpoints calculations).
with BANDS_LATTICE=AUTO aflows determines the lattice just before the static run

STATIC_BANDS OPTIONAL

A static run is performed with ad hoc INCARS. After the STATIC run an BANDS calculation is performed.

Look for STATIC_BANDS options for tuning the calculations.

You shall have a KPOINTS IMPLICIT and you must specify:

```
[VASP_KPOINTS_FILE]BANDS_LATTICE=fcc (cub,bcc,tet,bct,hex,orc,orci,orcc,orcf,rhl,tri)
[VASP_KPOINTS_FILE]BANDS_GRID=16 (grid, thickness of vasp kpoints calcaluations).
```

DIELECTRIC_STATIC (or DS)

If a static calculation is present (RELAX_STATIC=N,RELAX_STATIC_BANDS=N,STATIC_BANDS) then the keyword preceed by a comma will switch to DIELECTRIC_STATIC calculations operating on the INCAR, through LRPA, LEPSILON, etc... The delta_K of the grid is choesen to be

```
#define DIELECTRIC_DK 0.1 // aflow_ivasp.cpp
```

It works only with vasp version > 5.2.

Note that this drops the NPAR switch to allow k-points resampling so you must have a LOT of RAM, and set a large amount of stack ("ulimit -s unlimited"). If the code gets a MPICH error, it tries to circumvent it increasing the stack and decreasing the KPOINTS.

Example:

```
RELAX_STATIC_BANDS=2,DIELECTRIC_STATIC // OR ..,DS
```

will relax twice, perform the static calculations, get the bands, and run the static dielectric run.

DIELECTRIC_DYNAMIC (or DD)

If a static calculation is present (RELAX_STATIC=N,RELAX_STATIC_BANDS=N,STATIC_BANDS) then the keyword preceed by a comma will switch to DIELECTRIC_DYNAMIC calculations operating on the INCAR, through LRPA, LOPTICS, etc...

If not selected, it forces the DIELECTRIC_STATIC to be done before, as it requires the dielectric_static wavefunction

Example

```
RELAX_STATIC_BANDS=2,DIELECTRIC_STATIC,DIELECTRIC_DYNAMIC // OR ..,DS,DD
```

will relax twice, perform the static calculations, get the bands, and run the static dielectric and then the final dynamic dielectric calculation.

It works only with vasp version > 5.2.

REPEAT_BANDS OPTIONAL

This option is useful to repeat the band calculation after a previous relax*, static and band performance. All the relax and static is preserved.

The INCAR.bands POTCAR.bands, and POSCAR.bands are preserved while the KPOINTS is re-generated to allow different path. You can force this option by copying LOCK into a file REPEAT_BANDS so aflow finds a new directory to run (otherwise you shall delete LOCK and add the option to aflow.in).

REPEAT_STATIC_BANDS OPTIONAL

This option is useful to repeat the band calculation after a previous relax*, static and band performance. All the relax and static is preserved.

The INCAR.static POTCAR.static, and POSCAR.static are preserved while the KPOINTS is re-generated to allow different path. You can force this option by copying LOCK into a file REPEAT_STATIC_BANDS so aflow finds a new directory to run (otherwise you shall delete LOCK and add the option to aflow.in).

REPEAT_DELSOL OPTIONAL

This option is useful to repeat the delta-sol calculation after a previous static run. All *.static or *.static.bz2 files are preserved.

The POSCAR.static, KPOINTS.static, POTCAR.static are copied to POSCAR, KPOINTS, POTCAR for the delta-sol run. The INCAR.static is reread and modified to include NELECT tag.

N_0 = grep NELECT OUTCAR.static

Then delta_N is calculated using scheme from ref PRL 105,196403 (2010), two delta-sol runs are performed

(i) delta-sol plus (*.dsolp) with NELECT = N_0 + delta_N

(ii) delta-sol minus (*.dsolm) with NELECT = N_0 - delta_N

You can force this option by moving the LOCK file into a file named REPEAT_DELSOL so aflow finds a new directory to run

(otherwise you shall delete LOCK and add the option to aflow.in).

NOTE: If you specify more than one GENERATE/STATIC/KPOINTS/RELAX/RELAX_STATIC runs, the priority is GENERATE (1), STATIC (2), BANDS (3), KPOINTS (4), RELAX (5).

--- INPUT FILES -----

For INCAR,KPOINTS,POSCAR,POTCAR you must choose one of the available modes: EXPLICIT, IMPLICIT, EXTERNAL

*** INCAR *** ** INCAR *** ** INCAR *** ** INCAR ***

[VASP_INCAR_MODE_**PLICIT]

EXPLICIT and IMPLICIT mode of INCAR are supported.

In EXPLICIT mode everything on the left of "[VASP_INCAR_FILE]" strings is copied into INCAR files.

NOTE to [VASP_INCAR_MODE_EXPLICIT]

With [VASP_INCAR_MODE_EXPLICIT] activated, you can add:

[VASP_INCAR_MODE_EXPLICIT]START and

[VASP_INCAR_MODE_EXPLICIT]STOP

This helps because instead of specifying INCAR with the

"[VASP_INCAR_FILE]" strings, everything between the [...]START and [...]STOP keys are copied inside a INCAR.

This option is very useful trick to cut/paste long INCARs

without adding the "[VASP_INCAR_FILE]"

strings at the beginning of each line.

In IMPLICIT mode the keyword "[VASP_INCAR_FILE]SYSTEM_AUTO"

indicates aflow to take system, prototype and info names from the POSCAR (available if taken from the databases).

NOTE after the EXPLICIT and IMPLICIT constructions, aflow fixes the INCAR following the FORCE_OPTIONS keyword specified below.

[VASP_INCAR_MODE_EXTERNAL]

Searches a file or loads a file from stdout command:

[VASP_INCAR_FILE]FILE=../../somewhere/INCAR

or

[VASP_INCAR_FILE]COMMAND=bzcat ../somewhere/INCAR.relax2.bz2

if FILE or COMMAND are not specified, aflow takes the standard FILE=./INCAR as default.

*** KPOINTS *** ** KPOINTS *** ** KPOINTS *** ** KPOINTS ***

[VASP_KPOINTS_MODE_**PLICIT]

Both EXPLICIT and IMPLICIT are supported.

IMPLICIT: Everything after the line containing

[VASP_KPOINTS_FILE] will be used to create the KPOINTS file.

Options are

KMODE=X # MODE OF THE KPOINTS (same as second line of KPOINTS)

KPPRA=XXXX # number of kpoints times the size of unit cell
the grid is calculated with the function KPPRA

KSCHEME=Monkhorst-Pack (or Gamma)# Kpoints scheme. The ones of VASP are supported and you can use only 1 letter.

You can specify KSCHEME=AUTO (actually you need only =A)

and aflow takes 'G' for FCC/HEX lattices and 'M' otherwise

KSHIFT=X X X # the three shift to bring in/out the Gamma point
see manual

You can override the KPOINT generation only for the STATIC phase by enforcing:

STATIC_KMODE=X # see KPOINTS for explanation

STATIC_KPPRA=XXXX # see KPOINTS for explanation

STATIC_KSCHEME=Monkhorst-Pack# see KPOINTS for explanation

STATIC_KSHIFT=X X X # see KPOINTS for explanation

EXPLICIT: Everything after the line containing

[VASP_KPOINTS_MODE_EXPLICIT] will be used to generate the KPOINTS

file. Hence this string should be used once and just before the KPOINTS information.

NOTE to [VASP_KPOINTS_MODE_EXPLICIT]

With [VASP_KPOINTS_MODE_EXPLICIT] activated, you can add:

[VASP_KPOINTS_MODE_EXPLICIT]START and

[VASP_KPOINTS_MODE_EXPLICIT]STOP

This helps because instead of specifying KPOINTS with the "[VASP_KPOINTS_FILE]" strings, everything between the [...]START and [...]STOP keys are copied inside a KPOINTS. This option is very useful trick to cut/paste long KPOINTSs without adding the "[VASP_KPOINTS_FILE]" strings at the beginning of each line.

[VASP_KPOINTS_MODE_EXTERNAL]

Searches a file or loads a file from stdout command:

[VASP_KPOINTS_FILE]FILE=../../somewhere/KPOINTS

or

[VASP_KPOINTS_FILE]COMMAND=bzcat ../somewhere/KPOINTS.relax2.bz2

if FILE or COMMAND are not specified, aflow takes the standard FILE=./KPOINTS as default.

*** POSCAR *** *** POSCAR *** *** POSCAR *** *** POSCAR ***

[VASP_POSCAR_MODE_**PLICIT]

EXPLICIT and IMPLICIT modes of POSCAR are supported.

In EXPLICIT mode everything on the left of "[VASP_POSCAR_FILE]" strings is copied into POSCAR files.

NOTE to [VASP_POSCAR_MODE_EXPLICIT]

With [VASP_POSCAR_MODE_EXPLICIT] activated, you can add:

[VASP_POSCAR_MODE_EXPLICIT]START and

[VASP_POSCAR_MODE_EXPLICIT]STOP

This helps because instead of specifying POSCAR with the "[VASP_POSCAR_FILE]" strings, everything between the [...]START and [...]STOP keys are copied inside a POSCAR.

This option is very useful trick to cut/paste long POSCARs without adding the "[VASP_POSCAR_FILE]" strings at the beginning of each line.

[VASP_POSCAR_MODE_EXPLICIT]START.AAA

...

[VASP_POSCAR_MODE_EXPLICIT]STOP.AAA

[VASP_POSCAR_MODE_EXPLICIT]START.BBB

...

[VASP_POSCAR_MODE_EXPLICIT]STOP.BBB

aflow generates directories ARUN.AAA ARUN.BBB ... and runs them all with the same parameters but different POSCARs

In IMPLICIT mode, commands are given to generate structures.

The first must be the keyword "PROTOTYPE=", then we can have "SPECIES=" and "VOLUMES=". Keywords are separated by ";", while values are separated by ",".

PROTOTYPE=label identifies the label on the DMQC-HTQC or GUS database of prototypes.

SPECIES=specieA,specieB,... identifies the atomic species.

You can add the "_pv", "_sv".. etc which is used for the POTCAR atomic generation.

VOLUMES=volumeA,volumeB,... identifies the volume per atom of each specie and the overall volume of the cell is the sum of each individual volumes (Vegard's law).

If you specify only one volume, as

"VOLUME=volume", then the cell will be forced to have that volume per atom (aflow takes "volume" and multiply times the number of atoms in the cell).

If you do not specify any volume, then the CELL volume will be taken from the closed packed atomic volume (fcc through VASP) and averaged with the Vegard's law.

Example

[VASP_POSCAR_FILE]PROTOTYPE=5;VOLUMES=20,10;SPECIES=Ag,Zr_sv;

You can change/force the volume created by PROTOTYPE with

[VASP_POSCAR_FILE]VOLUME=xxx

[VASP_POSCAR_FILE]VOLUME+=xxx

[VASP_POSCAR_FILE]VOLUME*=xxx

[VASP_POSCAR_MODE_EXTERNAL]
Searches a file or loads a file from stdout command:
[VASP_POSCAR_FILE]FILE=../../somewhere/POSCAR
or
[VASP_POSCAR_FILE]COMMAND=bzcat ../somewhere/CONTCAR.relax2.bz2
if FILE or COMMAND are not specified, aflow takes the standard
FILE=./POSCAR as default.

*** POTCAR *** *** POTCAR *** *** POTCAR *** *** POTCAR ***
[VASP_POTCAR_MODE_**PLICIT] MANDATORY, no default
IMPLICIT: The "POTCAR" potential files specified after the
"[VASP_POTCAR_FILE]" strings are copied into POSCAR files.
If you specify the keyword "[VASP_POTCAR_FILE]SYSTEM_AUTO"
then aflow will extract the species names from the POSCAR
(stored inside the structure generation) adding before and after
the PREFIX and SUFFIX as:
[VASP_POTCAR_FILE]PREFIX=\$POTCARDIR/pot_LDA/current/
[VASP_POTCAR_FILE]SUFFIX=/POTCAR
This option is very powerful for automatic generation of
calculations.

EXPLICIT: Everything after the line containing
[VASP_POTCAR_MODE_EXPLICIT] will be used to generate the POTCAR
file. Hence this string should be used once and just before the
POTCAR information.

[VASP_POTCAR_MODE_EXTERNAL]
Searches a file or loads a file from stdout command:
[VASP_POTCAR_FILE]FILE=../../somewhere/POTCAR
or
[VASP_POTCAR_FILE]COMMAND=bzcat ../somewhere/CONTCAR.relax2.bz2
if FILE or COMMAND are not specified, aflow takes the standard
FILE=./POTCAR as default.

--- FORCE OPTIONS -----

[VASP_FORCE_OPTION]KEYWORD
Force some parameters for VASP calculation, changing the input files
appropriately. They are all OPTIONALS. Possible keywords are:

NOTUNE

Aflow/aflovd does not perform any modification of the input files so it
neglect all the FORCE_OPTIONS parameters.

SYSTEM_AUTO

Adapt INCAR adding the system, prototype and info names from
the POSCAR (available if taken from the databases).

STATIC

Adapt INCAR to perform a static run (IBRION,NSW,ISIF are commented).
You can mix RUN_RELAX and STATIC options to get particular behaviors.

RELAX || RELAX_ALL

Adapt INCAR to perform a relaxed run adapting (default MODE_ENERGY)
IBRION=2 # relax with Conjugate Gradient
NSW=51 # relax for long
ISIF=3 # relax everything
but without specifying the way to add.

RELAX_IONS

Adapt INCAR to perform a run in which only IONS are relaxed.

RELAX_CELL_SHAPE

Adapt INCAR to perform a run in which only CELL_SHAPE is relaxed.

RELAX_CELL_VOLUME

Adapt INCAR to perform a run in which only CELL_VOLUME is relaxed.

RELAX_IONS_CELL_VOLUME

Adapt INCAR to perform a run in which only IONS_CELL_VOLUME is relaxed.
Vasp does not support relaxation of IONS and VOLUME at the same time so
aflow runs alternate relaxations volume/ions (relaxODD/relaxEVEN). I suggest
to bump up N in [RELAX=N] so you achieve better convergence.

RELAX_MODE=ENERGY (DEFAULT)

Adapt INCAR to perform a relaxed run minimizing the total energy
IBRION=2 # relax with Conjugate Gradient
NSW=51 # relax for long
ISIF=3 # relax everything

RELAX_MODE=FORCES

Adapt INCAR to perform a relaxed run minimizing all the forces
NELMIN=4 # The forces have to be well converged
ADDGRID=.TRUE. # To support finer forces calculation
EDIFFG=-1E-5 # The final structure has to have zero forces!
IBRION=1 # More stable algorithm
NSW=100 # relax for very long
ISIF=3 # relax everything
If RELAX_MODE_FORCES and RELAX_MODE_ENERGY are both (or none) specified,
the default is to take RELAX_MODE_ENERGY.

PREC= LOW | MEDIUM | NORMAL | HIGH | ACCURATE), PRESERVED

PREC=LOW (PREC=LOW)

After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd re-adapts INCAR to enhance precision
ENMAX = XXXX # 1.0 ENMAX of pseudopotentials
PREC = low # reduce wrap around errors
Note: the 1.0 can be changed in aflow.h (VASP_PREC_ENMAX_MEDIUM) and recompiling.

PREC=MEDIUM (PREC=MEDIUM)

After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd re-adapts INCAR to enhance precision
ENMAX = XXXX # 1.3 ENMAX of pseudopotentials
PREC = med # reduce wrap around errors
Note: the 1.3 can be changed in aflow.h (VASP_PREC_ENMAX_MEDIUM) and recompiling.

PREC=NORMAL (PREC=NORMAL)

After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd re-adapts INCAR to enhance precision
ENMAX = XXXX # 1.3 ENMAX of pseudopotentials
PREC = normal # reduce wrap around errors
Note: the 1.3 can be changed in aflow.h (VASP_PREC_ENMAX_NORMAL) and recompiling.

PREC=ACCURATE (or PREC=HIGH or PREC=ACCURATE) (default=ACCURATE if not specified)

After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd re-adapts INCAR to enhance precision
ENMAX = XXXX # 1.4 ENMAX of pseudopotentials
PREC = Accurate # avoid wrap around errors
LREAL = .FALSE. # reciprocal space projection technique
EDIFF = 1E-6 # high accuracy required
ALGO = Fast # fast determination of ground state
Note: the 1.4 can be changed in aflow.h (VASP_PREC_ENMAX_HIGH) and recompiling.

PREC=PRESERVED

When AFLOW switches from relax to static and to bands, the PREC is preserved and not changed accordingly. Good
PREC conserved through the task.

ENMAX_MULTIPLY=NUMBER (default is VASP_PREC_ENMAX_LOW, _MEDIUM, _NORMAL, _HIGH, _ACCURATE see aflow.h)

Force the user choice of MULTIPLIER of the max_cutoff of pseudopotentials. Usually 1.2-1.4.
Increase to 1.5 or higher for high-pressure calculations.

ALGO=(NORMAL | VERYFAST | FAST | ALL | DAMPED), PRESERVED

ALGO=XXXXXX

After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd re-adapts INCAR to enforce ALGO=XXXXXX
ALGO and removing IALGO.

ALGO = XXXXXX # ALGO - XXXXXX
 XXXXX can be NORMAL | VERYFAST | FAST | ALL | DAMPED
 ALGO_PRESERVED
 When AFLOW switches from relax to static and to bands, the ALGO is preserved and not changed to "normal". Good relax correctly but crash on the static part.

METAGGA=TPSS | RTPSS | M06L | MBJL | SCAN | MSO | MS1 | MS2 | NONE
 METAGGA=XXXXXX
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd re-adapts INCAR to enforce METAGGA=
 METAGGA = XXXXXX # METAGGA = XXXXXX
 XXXXX can be TPSS | RTPSS | M06L | MBJL | SCAN | MSO | MS1 | MS2 | NONE
 If NONE or nothing is specified, the METAGGA keyword is not included in the INCAR.

IVDW=number_for_VASP_see_manual_for_IVDW
 IVDW=XXXXXX
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd re-adapts INCAR to enforce IVDW=XXX
 IVDW = XXXXXX # IVDW = XXXXXX
 XXXXX can be a number specified in https://cms.mpi.univie.ac.at/vasp/vasp/IVDW_approximate_vdW_correction_me
 If 0 or nothing is specified, the VDW keyword is not included in the INCAR.
 Note that other parameters might have to be specified in the INCAR part of aflow.in by the user

NBANDS
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd estimates the number of NBANDS and
 to the proper input files.
 NBANDS=XXXX
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd used XXXX as number of bands. This
 If the entry exists, it will override the original INCAR specification of the user.

PSTRESS=XXXX (in kB) (PRESSURE)
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings, aflow/aflowd adapts it for pressure calculations
 If the entry exists, it will override the original INCAR specification of the user.

NEGLECT_NOMIX
 The run is not performed if the system is known to be immiscible.
 The list of immiscibles are in aflow_nomix.cpp.

SPIN=ON
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
 aflow/aflowd re-adapts INCAR to include SPIN.
 If no SPIN is mentioned the INCAR spin part remains untouched.

SPIN=OFF
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
 aflow/aflowd re-adapts INCAR to exclude SPIN.
 If no SPIN is mentioned the INCAR spin part remains untouched.

SPIN=REMOVE_RELAX_1
 After 1 RELAXATION is performed, if there is no spin in the calculation,
 the spin is turned off automatically to save computer time and
 make relaxations easier. Default cutoff is 0.025 specified in aflow.h
 (VASP_SPIN_REMOVE_CUTOFF)

SPIN=REMOVE_RELAX_2
 After 2 RELAXATIONS are performed, if there is no spin in the calculation,
 the spin is turned off automatically to save computer time and
 make relaxations easier. Default cutoff is 0.025 specified in aflow.h
 (VASP_SPIN_REMOVE_CUTOFF)

BADER=ON | OFF (default OFF)
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
 aflow/aflowd re-adapts INCAR to include BADER analysis (LAECHG).
 It works only on the STATIC part of a run. RELAX_STATIC and RELAX_STATIC_BANDS
 have this option automatic.
 By specifying OFF, aflow will strip the INCAR of any BADER related instruction.

ELF=ON | OFF (default OFF)
 After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
 aflow/aflowd re-adapts INCAR to include the Electron Localization Function (ELF) analysis (LELF).

It works only on the STATIC part of a run.
By specifying OFF, aflow will strip the INCAR of any ELF related instruction.

LSCOUPLING=ON

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to include LSCOUPLING.

If you have good INCARs, you do not need to play with these options.
LSORBIT=.TRUE.

LNONCOLLINEAR=.TRUE.

It does not touch SAXIS = s_x s_y s_z so the default (0+,0,1) is kept,
unless you specify something different in the INCAR part of aflow.in
("0+" implies an infinitesimal small positive number in x direction).

With LSCOUPLING=ON MAGMOM is adapted as a vector for each atom.

When you're doing non-collinear calculations you have to specify a
vector for each atom, i.e. three entries per atom. So if you have N
ions you therefore will have to have 3N elements on the MAGMOM-line
which are the projections onto the chosen SAXIS.

NOTE: with LSORBIT and LNONCOLLINEAR calculations VASP must be compiled
without the option -DNGZhalf and -DNGXhalf.

AFLOW expects the executable to be called as "BIN"+"LS", therefore
if you specify "mpivasp46s", the LS run will be performed calling "mpivasp46sLS".

LSCOUPLING=OFF

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to exclude LSCOUPLING.

LSORBIT=.FALSE.

LNONCOLLINEAR=.FALSE.

MAGMOM is not touched and is left to be specified by AUTO_MAGMOM
or by the INCAR part of aflow.in.

AUTO_MAGMOM=ON

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to include AUTO_MAGMOM.

If you have good INCARs, you do not need to play with these options.

AUTO_MAGMOM=OFF

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to exclude AUTO_MAGMOM.

If you have good INCARs, you do not need to play with these options.

SYM=ON

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to include SYMMETRY (ISYM=2).

If you have good INCARs, you do not need to play with these options.

SYM=OFF

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to exclude SYMMETRY (ISYM=0).

If you have good INCARs, you do not need to play with these options.

KPOINTS=EVEN | ODD

Aflow makes KPOINTS even/odd.

KPOINTS=KSHIFT_GAMMA_EVEN | _ODD

Aflow shift of 1/2 the K points that are even/odd.

KPOINTS=GAMMA

Aflow sets [1 1 1] K points. You should specify the correct BINARY for the
the gamma point calculation (faster) otherwise aflow will run the normal vasp.

KPOINTS=KSCHEME_MONKHORST_PACK

Aflow forces Monkhorst-Pack Kscheme

KPOINTS=KSCHEME_GAMMA

Aflow forces Gamma Kscheme

KPOINTS=KSCHEME_AUTO

Aflow forces Kscheme to be Gamma for FCC/HEX lattices and Monkhorst-Pack for everything else.

KPOINTS=IBZKPT

Aflow uses IBZKPT.relax1 to proceed for relaxations >=2.

WAVECAR=ON

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,

aflow/aflowd re-adapts INCAR to include WAVECAR. (LWAVE = .TRUE.)
If you have good INCARs, you do not need to play with these options.
WAVECAR=OFF

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to exclude WAVECAR. (LWAVE = .FALSE.)
If you have good INCARs, you do not need to play with these options.

CHGCAR=ON

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to include CHGCAR. (LCHARG = .TRUE.)
If you have good INCARs, you do not need to play with these options.

CHGCAR=OFF

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to exclude CHGCAR. (LCHARG = .FALSE.)
If you have good INCARs, you do not need to play with these options.

RWIGS_STATIC

When running a STATIC calculation (STATIC, RELAX_STATIC, RELAX_BAND_STATIC)
it extracts RWIGS from the POTCAR (and forces LOREBIT=0)

TYPE=METAL or TYPE=INSULATOR or TYPE=SEMICONDUCTOR or TYPE=DEFAULT

Aflow arranges the integration method of the Brillouin Zone for Metals or
Insulators/Semiconductors by tuning ISMEAR and SIGMA. Look at the manual.
This key is very important for forces calculations (not really for bulk energies)
since the tetrahedron method with Blochl corrections is NOT variational
in the forces, so the answers might be wrong in some cases.

ISMEAR = 2 # default (hope)
SIGMA = 0.2 # default (hope)
ISMEAR = 1 # for metals
SIGMA = 0.1 # for metals
ISMEAR = 0 # for insulators/semiconductors
SIGMA = 0.05 # for insulators/semiconductors

This keyword is MANDATORY if you are doing phonons calculations.

LDAU1=ON | OFF | ADIABATIC | CUTOFF

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to include LDAU in the mode 1 of VASP.
You need to specify the species with the keyword.
With ADIABATIC it turns on LDAU adiabatically through the relaxN steps.
If n is the max number of relaxes and j is the step then the U and J parameters are
set as $j/n*(U,J)$ per calculation. The minimum step of ADIABATIC relaxations is set
up to be LDAU_ADIABATIC_RELAX_DEFAULT=6, and can be modified in aflow_kvasp.cpp
With CUTOFF, aflow adds an extra relaxation step (RELAX++). The extra step is performed
with a recycled CHGCAR of the previous step, static, and turning off all LDAU calculations.
This is very useful to calculate non LDAU energies with LDAU charge distributions.

LDAU2=ON | OFF | ADIABATIC | CUTOFF

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to include LDAU in the mode 2 of VASP.
You need to specify the species with the keyword.
With ADIABATIC it turns on LDAU adiabatically through the relaxN steps.
If n is the max number of relaxes and j is the step then the U and J parameters are
set as $j/n*(U,J)$ per calculation. The minimum step of ADIABATIC relaxations is set
up to be LDAU_ADIABATIC_RELAX_DEFAULT=6, and can be modified in aflow_kvasp.cpp
With CUTOFF, aflow adds an extra relaxation step (RELAX++). The extra step is performed
with a recycled CHGCAR of the previous step, static, and turning off all LDAU calculations.
This is very useful to calculate non LDAU energies with LDAU charge distributions.

To specify the parameters, you can have AFLOW to choose for you
with LDAU_SPECIES or do by hand through LDAY_PARAMETERS.

LDAU_SPECIES=Cu La O Ru (separated by spaces, no commas) so that aflow
picks the right parameters from the AVASP_Get_LDAU1_Parameters() routine (aflow_avasp.cpp).
If no LDAU_SPECIES are present, you have to specify LDAUL, LDAUU, LDAUJ manually
in the INCAR.

LDAU_PARAMETERS=speciesA,speciesB.;L_A,L_B.;U_A,U_B.;J_A,J_B...

for example (groups separated by ";" while entries separated by ",")
Cu,La,0,Ru;0,2,1,-1;0.0,4.2,1.1,0.0;0.0,0.0,0.0,0.0
where
Cu,La,0,Ru LDAU set of species
0,2,1,-1 LDAU orbitals with -1,0,1,2,3 = none,s,p,d,f
0.0,4.2,1.1,0.0 LDAU Us for species
0.0,0.0,0.0,0.0 LDAU Js for species
Note: LDAU_PARAMETERS overrides LDAU_SPECIES.

LDAU=OFF | LDAU1=OFF | LDAU2=OFF

After INCAR is generated by the "[VASP_INCAR_FILE]" strings,
aflow/aflowd re-adapts INCAR to exclude all LDAU calculations.

CONVERT_UNIT_CELL=STRING1,STRING2,... etc where STRINGS can be

STANDARD_PRIMITIVE | STD_PRIM | SPRIM

Converts the unit cell to the standard primitive form as described
by the rules in the aflow_kpoints.cpp file and in the README_LATTICE file
If specified, it turns off NIGGLI, MINKOWSKI, INCELL, COMPACT, WIGNERSEITZ.
If both STANDARD_PRIMITIVE and STANDARD_CONVENTIONAL then PRIMITIVE has priority.
REF: Setyawan Curtarolo, DOI: 10.1016/j.commatsci.2010.05.010

STANDARD_CONVENTIONAL | STD_CONV | SCONV

Converts the unit cell to the standard primitive form as described
by the rules in the aflow_kpoints.cpp file and in the README_LATTICE file
If specified, it turns off NIGGLI, MINKOWSKI, INCELL, COMPACT, WIGNERSEITZ.
If both STANDARD_PRIMITIVE and STANDARD_CONVENTIONAL then PRIMITIVE has priority.
REF: Setyawan Curtarolo, DOI: 10.1016/j.commatsci.2010.05.010

NIGGLI

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!).
(Written by Dane Morgan).

MINKOWSKI | MINK

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>

INCELL

Convert the basis with all atoms mapped to their images within
the unit cell.

COMPACT

Convert the basis with all atoms 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 complicated
molecules where atoms exit of the unit cell and you have
problems understanding where they are because visualization
packages do not show bonds anymore ...

WIGNERSEITZ | WS

Convert the basis with all atoms mapped to their images within
the Wigner Seitz cell.

CARTESIAN | CART

Convert the basis set to Cartesian coordinates.
FRACTIONAL | DIRECT | FRAC | DIR
Convert the basis set to fractional coordinates.
PRESERVE | PRES
Preserve the POSCAR from being Standardized (Primitive or Coventional).

VOLUME=xxx
VOLUME+=xxx
VOLUME*=xxx
Change the volume of the POSCAR accordingly (fix, +=, *) as in the c,c++ standard.

--- FROZSL OPTIONS -----

[AFLOW_FROZSL]CALC

Generate the POSCARS starting from the FROZSL input file (the long one you generate from the web). It uses a PERL script to parse the frozl part and then generates the poscars. Then it runs then, extract the energies, remove the minima, transform in hartree and pring everything in aflow.frozsl_energies.out
The FROZSL file is defined as everything after the [AFLOW_FROZSL]CALC line.

[AFLOW_FROZSL]DOWNLOAD | [AFLOW_FROZSL]DOWN

Starting from the
[FROZSL_MODE_EXPLICIT]START.FROZSL_STRUCTURE
xxx
[FROZSL_MODE_EXPLICIT]STOP.FROZSL_STRUCTURE
and, optionally,
[FROZSL_MODE_EXPLICIT]START.FROZSL_DIELECTRIC
xxx
[FROZSL_MODE_EXPLICIT]STOP.FROZSL_DIELECTRIC
download the FROZSL output by using "wget" and does the calculation as specified in the FROZSL code. The FROZSL code is saved in the aflow.frozsl_input.out file.

[FROZSL_MODE_PRESCRIPT]START
script to be run before starting FROZSL
[FROZSL_MODE_PRESCRIPT]STOP
[AFLOW_MODE_POSTSCRIPT]START
script to be run after starting FROZSL
[AFLOW_MODE_POSTSCRIPT]STOP

--- IGNORE OPTIONS -----

[VASP_FORCE_OPTION]IGNORE_AFIX=STRINGS

Activates/deactivates some parameters in aflow for VASP calculation, that are possible solutions to usual problems.
Suggestion: do not specify these options unless you are willing to look into OUTCAR and vasp.out to check for troubles.
They are all OPTIONALS. Possible keywords are:

IGNORE_AFIX=STRING1,STRING2,... etc where STRINGS can be

ROTMAT

"VERY BAD NEWS! Found some non-integer element in the rotation matrix" error. To address this issue, AFLOW removes symmetry (ISYM=0) and make K-points ODD (look at VASP tutorial about k-points).

SGRCON

SGRCON relaxation errors. By default aflow tries to go around SGRCON errors by enhancing SYMPREC=1e-6 in the INCAR.

IBZKPT

IBZKPT relaxation errors. By default aflow tries to go around IBZKPT errors by changing the KPOINTS to have origin in Gamma.

NKXYX_IKPTD

NK[X,Y,Z]>IKPTD kpoints errors. By default aflow tries to go around NK[X,Y,Z]>IKPTD errors by reducing the KP

SYMPREC

SYMPREC relaxation errors. By default aflow tries to go around SYMPREC errors by increasing precision with SYMPREC.

INVGRP

INVGRP relaxation errors. By default aflow tries to go around INVGRP errors by increasing precision with SYMPREC.

EDDRMM

EDDRMM relaxation errors. By default aflow tries to go around EDDRMM errors by changing the KPOINTS to have origin in Gamma.

LREAL

REAL_OPTLAY (1) errors. By default aflow tries to go around LREAL errors by changing the INCAR.

BRMIX

BRMIX errors. By default aflow tries to go around the BRMIX error problem by changing INCAR schemes.

DAV

DAV relaxation errors. By default aflow tries to go around DAV relaxation errors by changing INCAR schemes.

EFIELD_PEAD

EFIELD_PEAD errors. By default aflow tries to go around EFIELD_PEAD errors by dividing by 5 the EFIELD_PEAD.

EDDDAV

EDDDAV errors. By default aflow tries to go around EDDDAV errors by changing INCAR schemes.

ZPOTRF

ZPOTRF errors. By default aflow tries to go around ZPOTRF errors by changing INCAR schemes and potentially PO

EXCCOR

EXCHANGE-CORRELATIONS errors. By default aflow tries to go around "supplied exchange-correlation table is too errors by changing the POSCAR volume (inflating it).

NATOMS

NEAR NEAREST ATOMS errors. By default aflow tries to go around "The distance between some ions is very small" errors by changing the POSCAR volume (inflating it)

NBANDS

NBANDS errors. By default aflow tries to go around insufficient NBANDS by restarting VASP with increasingly higher NBANDS until everything is set. This can be done by tuning the INCAR schemes.

MEMORY

MEMORY errors. By default aflow tries to go around insufficient MEMORY by skipping the calculation and writing a SKIP file with some information inside.

PSMAXN

PSMAXN errors. By default aflow tries to go around PSMAXN warnings by restarting VASP with reducingly lower ENMAX until everything is set. This can be done by tuning the INCAR schemes.

NPAR

NPAR errors. By default aflow tries to go around NPAR warnings by restarting VASP with reducingly lower NPAR. This can be done by tuning the INCAR schemes.

NPARC

NPAR=number of cores. By default aflow tries to go around NPAR warnings by putting NPAR=4 (default).

NPARN

NPAR=number of nodes errors. By default aflow tries to go around NPAR warnings by putting NPAR=4 (default).

NPAR_REMOVE

NPAR=number of nodes errors, when VASP wants to change NPAR. By default aflow tries to go around NPAR warning

CSLOSHING

CSLOSHING electronic charge-sloshing problems. By default aflow tries to go around unconverged electronic loop due to charge sloshing, by restarting VASP with with different relax algorithm (aflow choses Algo=Normal which

contains default parameters better suited to address this issue).

DENTET

DENTET warnings. By default aflow tries to go around DENTET warnings by restarting VASP with different smearing algorithm. This can be done by tuning the INCAR schemes.

LRF_COMMUTATOR

LRF_COMMUTATOR warnings. By default aflow tries to go around LRF_COMMUTATOR warnings by restarting VASP with different options. This can be done by tuning the INCAR schemes.

GAMMA_SHIFT

GAMMA_SHIFT warnings. By default aflow tries to go around GAMMA_SHIFT warnings by restarting VASP with different options. This can be done by tuning the INCAR schemes.

MPICH11

MPICH11 errors. By default aflow tries to go around MPICH11 errors by restarting VASP without NPAR in INCAR, etc.

MPICH139

MPICH139 errors. By default aflow tries to go around MPICH139 errors by restarting VASP reducing KPOINTS and etc.

READ_KPOINTS_RD_SYM

READ_KPOINTS_RD_SYM warnings. By default aflow tries to go around READ_KPOINTS_RD_SYM warnings by restarting VASP with different options. This can be done by tuning the INCAR schemes.

<-----cut here----->

EXAMPLE

<-----cut here----->

```
[AFLOW] *****
[AFLOW]
[AFLOW]          .o.          .o88o. oooo
[AFLOW]          .888.          888 ' ' '888
[AFLOW]          .8'888.      o888oo 888 .ooooo. oooo oooo  ooo
[AFLOW]          .8' '888.      888 888 d88' '88b '88. '88. .8'
[AFLOW]          .88ooo8888.    888 888 888 888 '88..]88..8'
[AFLOW]          .8' '888.      888 888 888 888 '888' '888'
[AFLOW]          o88o  o888o o888o  o888o 'Y8bod8P' '8' '8' .in
[AFLOW]
[AFLOW] *****
[AFLOW] * Stefano Curtarolo - (aflow V30348)
[AFLOW] * Dane Morgan - Wahyu Setyawan - Gus Hart - Michal Jahnatek - Shidong Wang - Ohad Levy
[AFLOW] *****
[AFLOW] Aflow automatically generated (aflow_avasp.cpp)
[AFLOW] *****
[AFLOW] SYSTEM=Si1_ICSD_67788
#[AFLOW] single element calculation
[AFLOW] *****
[AFLOW] input file for aflow
[AFLOW_MODE=VASP]
[AFLOW] *****
[AFLOW_MODE_ZIP=bzip2]
[AFLOW_MODE_BINARY=vasp46s]
[AFLOW] *****
[AFLOW] *****
#[AFLOW_MODE_MPI]
[AFLOW_MODE_MPI_MODE] NCPUS=MAX
[AFLOW_MODE_MPI_MODE] COMMAND ="mpirun -np"
[AFLOW_MODE_MPI_MODE] AUTOTUNE
[AFLOW_MODE_MPI_MODE] BINARY="mpivasp46s"
[AFLOW] *****
[AFLOW_SYMMETRY] CALC
#[AFLOW_SYMMETRY] SGROUP_WRITE
#[AFLOW_SYMMETRY] SGROUP_RADIUS=7.77
[AFLOW] *****
#[AFLOW_NEIGHBOURS] CALC
```

```

[AFLOW_NEIGHBOURS] RADIUS=7.7
[AFLOW_NEIGHBOURS] DRADIUS=0.1
[AFLOW] *****
#[AFLOW_APL] CALC // README_AFLOW_APL.TXT
[AFLOW_APL] ENGINE=DM // README_AFLOW_APL.TXT
[AFLOW_APL] DISMAG=0.015 // README_AFLOW_APL.TXT
[AFLOW_APL] MINATOMS=100 // README_AFLOW_APL.TXT
#[AFLOW_APL] SUPERCELL=3x3x3 // README_AFLOW_APL.TXT
[AFLOW_APL] DC=y // README_AFLOW_APL.TXT
[AFLOW_APL] DPM=y // README_AFLOW_APL.TXT
[AFLOW_APL] ZEROSTATE=y // README_AFLOW_APL.TXT
[AFLOW_APL] DOS=y // README_AFLOW_APL.TXT
[AFLOW_APL] TP=y // README_AFLOW_APL.TXT
[AFLOW_APL] TPT=0:2000:10 // README_AFLOW_APL.TXT
[AFLOW] *****
#[AFLOW_QHA] CALC // README_AFLOW_APL.TXT
[AFLOW_QHA] GP_VOL_DISTORTION_PERCENTAGE=0.03 // README_AFLOW_APL.TXT
[AFLOW_QHA] DISPLACEMENTS=y // README_AFLOW_APL.TXT
[AFLOW_QHA] PROJECTION_DIR=1:1:1 // README_AFLOW_APL.TXT
[AFLOW_QHA] EOS=n // README_AFLOW_APL.TXT
[AFLOW_QHA] EOS_VOLRANGE_DIST=-2:4:0.5 // README_AFLOW_APL.TXT
[AFLOW_QHA] EOS_KPOINTS_MODE=32768:10000:20:100000 // README_AFLOW_APL.TXT
[AFLOW] *****
#[AFLOW_AAPL] CALC // README_AFLOW_APL.TXT
[AFLOW_AAPL] TDMAG=0.015 // README_AFLOW_APL.TXT
[AFLOW_AAPL] CUT_SHELL=4 // README_AFLOW_APL.TXT
[AFLOW_AAPL] CUT_RAD=4.5 // README_AFLOW_APL.TXT
[AFLOW_AAPL] SUMRULE=1E-5 // README_AFLOW_APL.TXT
[AFLOW_AAPL] BTE=FULL // README_AFLOW_APL.TXT
[AFLOW_AAPL] THERMALGRID=21x21x21 // README_AFLOW_APL.TXT
[AFLOW_AAPL] ISOTOPE=y // README_AFLOW_APL.TXT
[AFLOW_AAPL] CUMULATIVEK=y // README_AFLOW_APL.TXT
[AFLOW_AAPL] BOUNDARY=n // README_AFLOW_APL.TXT
[AFLOW_AAPL] NANO_SIZE=100 // README_AFLOW_APL.TXT
[AFLOW_AAPL] TCT=200:700:20 // README_AFLOW_APL.TXT
[AFLOW] *****
[VASP_RUN] RELAX=2 // GENERATE STATIC RELAX=N RELAX_STATIC=N STATIC_BANDS RELAX_STATIC_BANDS
#[VASP_FORCE_OPTION] NEGLECT_NOMIX
[VASP_FORCE_OPTION] CHGCAR=OFF // ON | OFF (default ON)
[VASP_FORCE_OPTION] KPOINTS=KEEPK
#[VASP_FORCE_OPTION] KPOINTS=EVEN // EVEN | ODD (default none)
#[VASP_FORCE_OPTION] KPOINTS_KSHIFT_GAMMA=EVEN // EVEN | ODD (default none)
#[VASP_FORCE_OPTION] KPOINTS_KSCHEME=MONKHORST_PACK // MONKHORST_PACK | GAMMA (manual)
#[VASP_FORCE_OPTION] KPOINTS_GAMMA
#[VASP_FORCE_OPTION] KPOINTS_IBZKPT
[VASP_FORCE_OPTION] SYM=ON // ON | OFF (default ON)
[VASP_FORCE_OPTION] AUTO_PSEUDOPOTENTIALS=potpaw_PBE // pot_LDA | pot_GGA | potpaw_LDA | potpaw_GGA | potpaw_PBE
[VASP_FORCE_OPTION] NBANDS // Estimate Bands (better than VASP)
#[VASP_FORCE_OPTION] SPIN=ON, REMOVE_RELAX_1 // (ON | OFF (default ON)), REMOVE_RELAX_1 | _2
#[VASP_FORCE_OPTION] AUTO_MAGMOM=ON // ON | OFF (default OFF)
[VASP_FORCE_OPTION] RELAX_MODE=FORCE // ENERGY | FORCES | ENERGY_FORCES | FORCES_ENERGY (default ENERGY)
[VASP_FORCE_OPTION] PREC=ACCURATE // (LOW | MEDIUM | NORMAL | HIGH | ACCURATE), PRESERVED (default NORMAL)
[VASP_FORCE_OPTION] ALGO=NORMAL // (NORMAL | VERYFAST | FAST | ALL | DAMPED), PRESERVED (default NORMAL)
[VASP_FORCE_OPTION] RELAX
#[VASP_FORCE_OPTION] NOTUNE
[VASP_FORCE_OPTION] TYPE=INSULATOR // METAL | INSULATOR | SEMICONDUCTOR | DEFAULT (default METAL)
#[VASP_FORCE_OPTION] CONVERT_UNIT_CELL=something // SPRIM, SCONV, NIGGLI, MINK, INCELL, COMPACT, WS, CART, etc.
#[VASP_FORCE_OPTION] VOLUME+=10.0
#[VASP_FORCE_OPTION] VOLUME*=1.05
[AFLOW] *****
[AFLOW] *****
[VASP_INCAR_MODE_EXPLICIT] START
SYSTEM=Si1_ICSD_67788
NELM = 120
NELMIN=2

```

```

LPLANE=.TRUE.
LREAL=.FALSE.
LSCALU=.FALSE.
PSTRESS=000      # in kBar (1kB=0.1GPa)          # for hand modification
#NBANDS=XX       # for hand modification
#IALGO=48        # for hand modification
[VASP_INCAR_MODE_EXPLICIT]STOP
[AFLOW] *****
[VASP_KPOINTS_MODE_IMPLICIT]
[VASP_KPOINTS_FILE]KSCHEME=M
[VASP_KPOINTS_FILE]KPPRA=3456
[VASP_KPOINTS_FILE]STATIC_KSCHEME=M
[VASP_KPOINTS_FILE]STATIC_KPPRA=3456
[VASP_KPOINTS_FILE]BANDS_LATTICE=FCC
[VASP_KPOINTS_FILE]BANDS_GRID=20
[AFLOW] *****
[AFLOW] *****
[VASP_POSCAR_MODE_EXPLICIT]START
Si1 #216 - (Si1_ICSD_67788) - Si1 [Si1] cF8 F-43m Si 216 Si1_ICSD_67788 ICSD_67788 (icsd library) (WICKOFF 216
1.224745
    0.00000000000000    2.20127478218115    2.20127478218115
    2.20127478218115    0.00000000000000    2.20127478218115
    2.20127478218115    2.20127478218115    0.00000000000000
2
Direct(2) [A2]
    0.00000000000000    0.00000000000000    0.00000000000000 Si
    0.25000000000000    0.25000000000000    0.25000000000000 Si
[VASP_POSCAR_MODE_EXPLICIT]STOP
[AFLOW] *****
[VASP_POTCAR_MODE_IMPLICIT]
[VASP_POTCAR_FILE]Si
[AFLOW] potpaw_PBE: Si
[AFLOW] COMPOSITION_PP=|Si2|
[AFLOW] COMPOSITION=|Si2|
[AFLOW] *****
[AFLOW] Aflow automatically generated (aflow_vasp.cpp)
[AFLOW] *****
<-----cut here----->

*****
*
*           aflow - STEFANO CURTAROLO Duke University 2003-2017
*           High-Throughput ab-initio Computing Project
*
*****

```