Basin Hopping VASP
Instruction Manual

January 2020

This is a C/C++/Bash implementation of a Basin Hopping Monte Carlo algorithm for the search of metal clusters global minima, coupled with VASP for DFT calculations. This implementation works with supported and gas phase clusters.

1. How to install

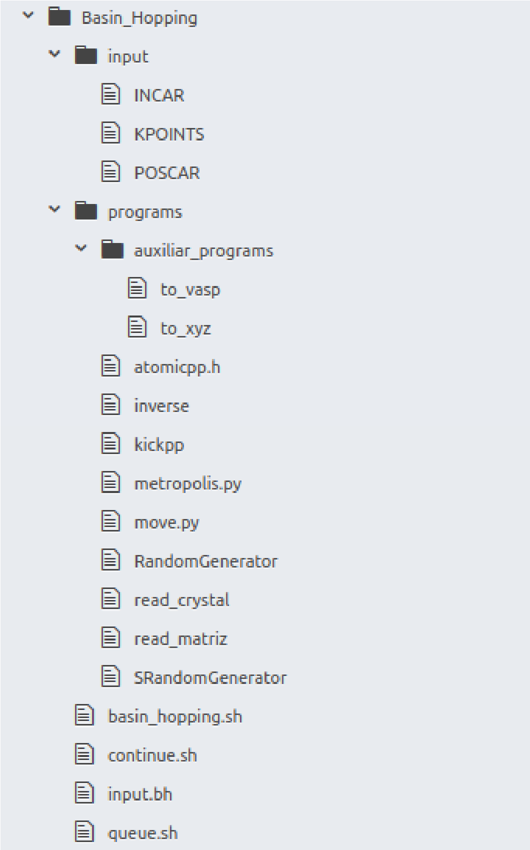
Download or copy and paste the *install.sh* file in your work directory. Make sure you have a recent version of gcc compiler (at least c++11) and Python3.6. Change permissions with chmod and execute it. This will create the directories and files necessaries and will compile it. So is not necessary to download any file more from the Github repository.

```
chmod +x install.sh
./install.sh
```

2. How to use

2.1. Summary of contents

input.bh contains parameters and variables for the control of the Basin Hopping algorithm. *queue.sh* file is a running and scheduling script. *basin_hopping.sh* is the main program. *continue.sh* restart the search, is the second part of the main program. *input* directory contains the inputs needed for VASP. *programs* contains the source code, the executable binaries, some temporal files produced during program execution and two helpful tools: *to_xyz* allows you convert VASP format to xyz. And *to_vasp* allows you convert xyz format to VASP. More about this in their respectives sections.



```
Basin_Hopping
    input
        INCAR
        KPOINTS
        POSCAR
    programs
        auxiliar_programs
            to_vasp
            to_xyz
        atomicpp.h
        inverse
        kickpp
        metropolis.py
        move.py
        RandomGenerator
        read_crystal
        read_matriz
        SRandomGenerator
    basin_hopping.sh
    continue.sh
    input.bh
    queue.sh
```

2.2. POTCAR

Will be created automatically.

2.3. INCAR

Is the usual INCAR VASP file. Here an example.

```
#DFT calculation
 ISTART = 0
 ICHARG = 2        # charge: 1-file 2-atom 10-const
 ISPIN  = 2        # spin polarized calculation (2-yes 1-no)
 INIWAV = 1        # random initialization for wf.

#Paralelization...depends on your system
 NCORE   = 32



#Electronic Relaxation
 ENCUT  = 500
 NELM   = 40       # maximum of 40 electronic steps
 EDIFF  = 1E-4     # accuracy for electronic minimization
 PREC   = Normal    # Normal or Accurate

#Ionic relaxation
 IBRION = 2
 ISIF   = 2           # relax ions only
 NSW    = 500
 POTIM  = 0.5

#Real space projection
 LREAL = .FALSE.       # projections done in real space

#Smearing
 ISMEAR =  1
 SIGMA  =  0.01

#Wavefunctions
 LWAVE  = .FALSE.
 LCHARG = .FALSE.
 LVTOT  = .FALSE.

#Dispersion
 IVDW   =  21
#Dipolar correction
 IDOPOL = 1
```

2.4. KPOINTS

Is the usual KPOINTS file. Gamma is shown below.

```
stnd (Automatic mesh)
0
Monkhorst Pack
  1  1  1
  0  0  0
```

2.5. to_vasp

The next step is to create an adecuate POSCAR. You may have already atomic coordinates of your system in *.xyz format. *to_vasp* can help you. This will be specially useful if you use Selective Dynamics. A file with Quantum Espresso format will be converted into a POSCAR file. However the xyz format has no information about the cell as VASP. In the example below the left column is a *.xyz file with selective dynamics as in Quantum Espresso, *to_vasp* convert it into VASP format except for the lattice vectors (right column), so you have to write it by hand to complete the file.

```
192                                      Ti   O
                                         64   128
Ti 2.962999 0.000000 -8.116570 0 0 0     Selective Dynamics
Ti 2.963001 6.563365 -8.116569 0 0 0     Cartesian
Ti 5.926000 0.000000 -8.116569 0 0 0      2.96299   0.00000  -8.11656    F  F  F
Ti 5.926000 6.563365 -8.116568 0 0 0      2.96300   6.56336  -8.11656    F  F  F
Ti 8.889001 0.000000 -8.116570 0 0 0      5.92600   0.00000  -8.11656    F  F  F
Ti 8.888998 6.563365 -8.116569 0 0 0      5.92600   6.56336  -8.11656    F  F  F
Ti 0.000000 3.281690 -4.288447            8.88900   0.00000  -8.11656    T  T  T
Ti 0.000000 9.845040 -4.288447            8.88899   6.56336  -8.11656    T  T  T
Ti 2.963000 3.281690 -4.288451            0.00000   3.28168  -4.28844    T  T  T
Ti 2.963000 9.845040 -4.288451            0.00000   9.84504  -4.28844    T  T  T
Ti 5.926000 3.281689 -4.288447            2.96300   3.28168  -4.28845    T  T  T
Ti 5.926000 9.845041 -4.288447            2.96300   9.84504  -4.28841    T  T  T
Ti 8.889000 3.281690 -4.288451            1.48149   5.28059  -7.90121    T  T  T

             . . .                                    . . .
```

Use *to_vasp* as every other command in bash (like cp or mv). The program will read *inputfile* and will write *outputfile*.

```
    to_vasp inputfile outputfile
```

If you prefer you can show the output in display by writting

```
    to_vasp inputfile
```

Or you can redirect the output to an existing file

```
    to_vasp inputfile >> existingfile
```

2.6. to_xyz

*.xyz format is supported by most visualization software. You can convert the CONTCAR or POSCAR files either in Direct or Cartesian coordinates to *.xyz format by using *to_xyz*. This program uses the lattice vectors to transform the direct coordinates to cartesian. It will recognize the format automatically. The program will read *inputfile* and will write *outputfile*.

```
to_xyz inputfile outputfile
```

Show the output in display by writting

```
to_xyz inputfile
```

Redirect the output to an existing file

```
to_xyz inputfile >> existingfile
```

2.7. POSCAR

Contains the information about the supercell and the atomic coordinates. Depending on if the system is supported or not (gas phase) the POSCAR will need a different format.

2.7.1. Gas phase

Only the first 5 lines are needed. The other lines will be generated automatically by the program.

```
BH_DFT
1.00000000000000
      11.851999999997867    0.0000000000000000    0.0000000000000000
      0.0000000000000000    13.126700000001802    0.0000000000000000
      0.0000000000000000    0.0000000000000000    29.999996692806895
```

2.7.2. Supported

The whole POSCAR file of the crystal is needed, either in Direct or Cartesian format.

```
BH_DFT
1.00000000000000
      11.851999999997867    0.0000000000000000    0.0000000000000000
      0.0000000000000000    13.126700000001802    0.0000000000000000
      0.0000000000000000    0.0000000000000000    29.999996692806895
Ti   O
64   128
Selective Dynamics
Cartesian
 0.000000000   3.281689681 -11.003424983    F  F  F
 0.000000000   9.845040604 -11.003424983    F  F  F
 2.963000053   3.281689476 -11.003424940    F  F  F
 2.963000053   9.845040809 -11.003424940    F  F  F
                          .  .  .
```

2.8. input.bh

This file contains the basin hopping parameters. Edit with nano or vi.

```
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
/_/_/_/_/_/_/_/_/_/_/_/_/_/ Basin Hopping  input file /_/_/_/_/_/_/_/_/_/_/_/_/
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/



initialization_file = /path/to/file # initialice the calculations starting
                                    # from that file (must be in POSCAR format)
                                    # Write ’false’ instead of a
                                    # path to  initialice from scratch

pseudo_dir = /path/to/pseudos       # In order to create POTCAR file is
                                    # needed the absolute path to the
                                    # directory with the VASP pseudopotentials.

pseudo_type =                       # Blank for PBE. If you want to use the
                                    # GW Gold VASP pseudopotential (Au_GW)
                                    # just write pseudotype=_GW

file_name = Au20_Rutile             # The program will create a directory
                                    # as output


cluster_ntyp = [Au:2]               # [Type1:Number]; [Type2:Number]
                                    # Write just one for monometallic clusters
                                    # and both for bimetallic.

temperature_K = 2500                # Parameter for metropolis criterion

iterations = 100                    # Iterations or basin hopping steps

randomness = 1                      # Select a random generator
                                    # 1-fully random    0-not so random

kick = 1                            # Select a kick algorithm
                                    # 1-kick with potential  0-just kick

step_width = 1.5                    # How strong is the kick
                                    # 1.5 for kick = 1  0.8 for kick = 0
```

2.9. Running

Once you wrote the input files modify the *queue.sh*, The next script has been written to work with LSF scheduler. The important is to run directly *basin_hopping.sh*, the parallelization (with MPI) is made automatically. The script takes as argument a file named input.bh by default. However you can write several input files with different names in order to run simultaneously several jobs with different systems.

Some computers have limited time per job. So it's possible to doesn't finish the whole process. If the job exceeds that time the computer will kill it. If the main program didn't finish the global optimization run *continue.sh* to restart the search from the last iteration. *continue.sh* needs the input.bh file as argument. *N.B.* The *file_name* must be the same that the used previusly. Itself recognize the last step.

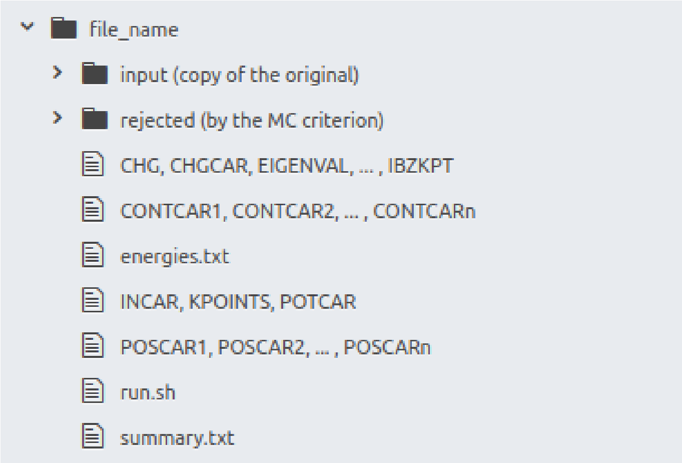
```
#!/bin/bash
#BSUB -q q_hpc
#BSUB -oo output
#BSUB -eo error
#BSUB -n 32
module load use.own
module load vasp/5.4.4
#BSUB -R "span[ptile=32]"
#BSUB -m "g3_a g3_b"
#BSUB -J Job_Name


#./continue.sh input.bh 2> ERRORES
./basin_hopping.sh input.bh 2> ERRORES > OUTPUT
```

Comment *continue.sh* to run the main program. Or comment *basin_hopping.sh* to run *continue.sh* and finally queue the script.

```
    bsub < queue.sh
```

2.10. Output

A new directory will be created. This will contain all the output. Name it as you want with the variable *file_name* in *input.bh* The VASP output files such as OUTCAR, EIGENVAL, etc, will be overwritted continually. INCAR, KPOINTS and POTCAR still in this directory along all the process. POSCAR and CONTCAR changes each iteration, however these files may be important. For this reason we need to save those files. After a step are created POSCARn and CONTCARn.



file_name
- input (copy of the original)
- rejected (by the MC criterion)
- CHG, CHGCAR, EIGENVAL, ... , IBZKPT
- CONTCAR1, CONTCAR2, ... , CONTCARn
- energies.txt
- INCAR, KPOINTS, POTCAR
- POSCAR1, POSCAR2, ... , POSCARn
- run.sh
- summary.txt

POSCAR has the initial coordinates and CONTCAR the relaxed ones. e.g. POSCAR20 and CONTCAR20 are the result of the iteration number 20. *input* is a copy of the original *input* directory. *rejected* contains all the coordinates rejected by the Monte Carlo criterion. *energies.txt* is a summary of all the energies and coordinates obtained. *sorted.txt* is also a summary but with energies sorted to highest to lowest. So the first in this file is the minimum founded. If the minimum founded is the 15th step then the corresponding file of coordinates is CONTCAR15. When the whole process is over is generated *summary.txt* with all the information about the parameters and variables for VASP and for the BH algorithm that were used.

3. Notes and recommendations

randomness=1 is recommended for small clusters (< 10 atoms) and randomness=0 for larger. kick=1 is recommended for supported clusters and for large clusters in gas phase (> 5 atoms). kick=0 is recommended only for very small ones (< 5 atoms) in gas phase.

In order to lay the cluster near of the surface we need to know the maximum z-coordinate of the crystal. The program recognizes it automatically. By periodicity of boundary conditions it is irrelevant for DFT calculations if the coordinates of some atoms are out of the simulation cell. However it is very import for the algorithm that the crystall coordinates must be all contained in the unit cell in one piece.(See Fig. 1)

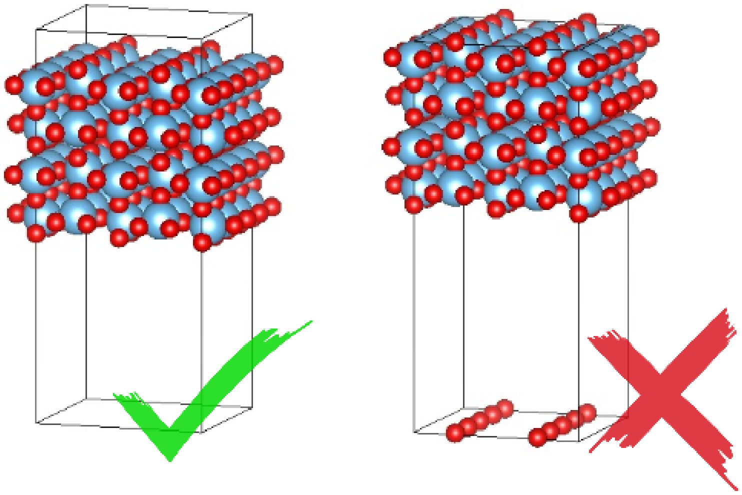


Fig. 1.