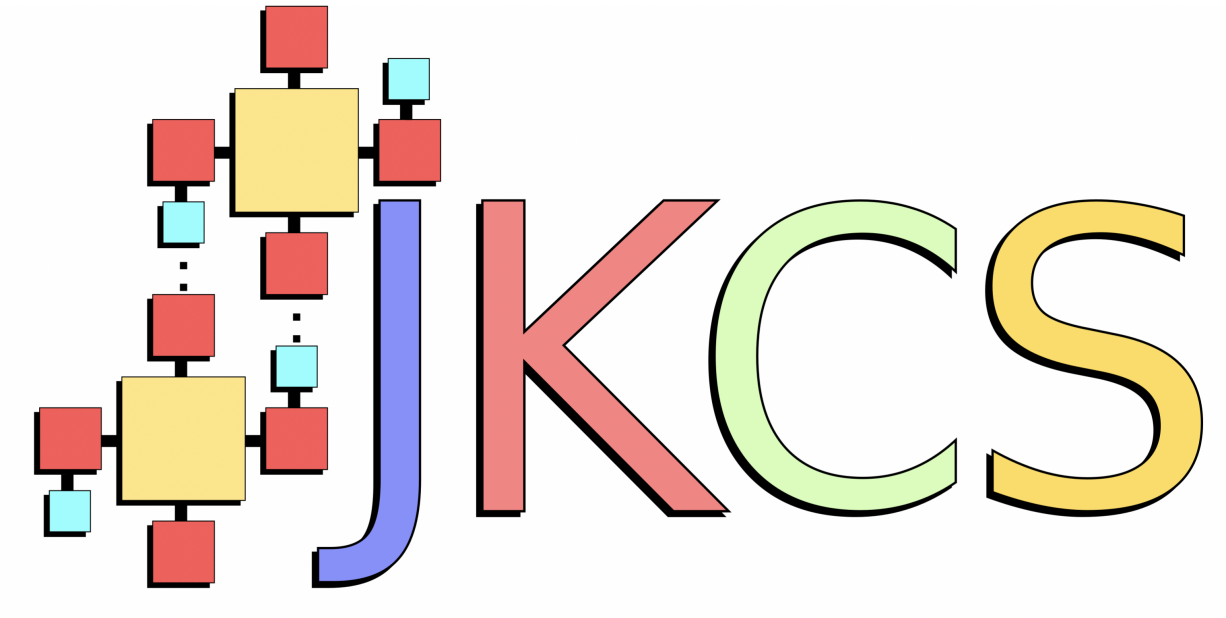


JKCS2.1 - MANUAL

Jammy Key
for
Configurational Sampling

Jakub Kubečka

in collaboration with:
Vitus Besel, Ivo Neefjes

thanx also to:
Fatemeh Keshavarz, Matias Jääskeläinen and Antti Toropainen

December 21, 2020

Contents

Introduction

JKCS presents a systematic approach to configurational sampling of molecular clusters. To perform configurational sampling on a chosen cluster, JKCS starts from the individual monomers that comprise the cluster. Of these monomers, all possible conformers are taken into account. This is done because the optimal structure of an individual monomer is not necessarily the optimal structure for this monomer inside the cluster. Also relevant deprotonated and protonated conjugates of monomers are considered, as proton transfer can occur between components in the cluster. JKCS will find all possible combinations of these monomer conformers that fulfil the chosen cluster composition and total charge. Of each of these combination, a large number of guess structures are created by adding the conformers together at random orientations. The guess structures of all possible combinations are subsequently optimized through quantum chemistry calculations at progressively higher levels of theory. After each optimization, unlikely structures are removed. The filtering out of these structures allows the next calculation at a higher level of theory to run at a reasonable computation cost. At the end of the configurational sampling, we end up with just a few appropriate structures that are optimized at a high level of theory.

It is important to note that there is no guarantee that the global minimum of the cluster is actually found through configurational sampling. It becomes more probable if we include all monomer conformers, use a large number of guess structures and optimize more structures at higher levels of theory, but absolute certainty that the global minimum is found will never be achieved.

JKCS PROVIDES

- a collection of scripts which can be used for configurational sampling with a large set of structures and files
- communication with 3rd-party computational programs such as Gaussian, ORCA, ABCluster and XTB
- automated handling of jobs and submissions to (super)computer clusters

HOW TO INSTALL or TEST?
Either follow to the next chapter or see the following videos:

- INSTALLATION: https://youtu.be/9mw45tbj1G4
- VERY FAST SHOW OF JKCS: https://youtu.be/xKKWZrO-EmU
- USING JKCS: https://youtu.be/C4dAkhU7O8E

HOW TO CITE?
At the moment, there is only an article about configurational sampling written by Kubečka *et al.* [3] to cite. An article about the program itself is currently in the submission process.

JKCS2.1

- most bugs are removed
- many parts are accelerated
- GitHub version (https://github.com/kubeckaj/JKCS2.1.git)
- almost proper manual :-D
- user-friendly coding (see example in Figure 1)

```
USER:$ JKCS0_copy SA A
 JKCS0_copy: File input.txt has been copied to your directory.
 JKCS0_copy: Change it (e.g., with program vim) and then run JKCS1_prepare.
USER:$ JKCS1_prepare
 JKCS1_prepare: Folder SYS_1SA_1A has been prepared.
 JKCS1_prepare: Folder SYS_2SA_2A has been prepared.
 JKCS1_prepare: DONE :-) --> Total number of cluster types:          2
 JKCS1_prepare: DONE :-) --> Total number of monomeric combinations: 11
 JKCS1_prepare: Now, just run JKCS2_explore.
USER:$ JKCS2_explore -pop 500 -gen 100
 JKCS2_explore:   All subfolders = SYS_1SA_1A SYS_2SA_2A
 JKCS2_explore: I, this script, will enter to all subfolders. :-D
 JKCS2_explore: Entering directory SYS_1SA_1A
 JKCS2_explore: ABCluster parameters set: LM(200),GEN(100),INIT(500),SC(4),BOX(9).
 JKCS2_explore:   The job has been submitted (24286472)
                  :
```

Figure 1: Example of the user-friendly coding.

PROGRAM USAGE

The JKCS program has already been utilized in several works. See examples of sampled systems in references [1, 2, 3, 4, 5].

CITATION

[1] Ahonen, L., Li, C., Kubečka, J., Iyer, S., Vehkamäki, H., Petäjä, T., Kulmala, M., and Hogan Jr, C. J. Ion mobility-mass spectrometry of iodine pentoxideâĂŞiodic acid hybrid cluster anions in dry and humidified atmospheres. *J. Phys. Chem. Lett. 10*, 8 (2019), 1935–1941.

[2] Keshavarz, F., Shcherbacheva, A., Kubečka, J., Vehkamäki, H., and Kurtén, T. Computational study of the effect of mineral dust on secondary organic aerosol formation by accretion reactions of closed-shell organic compounds. *J. Phys. Chem. A 123*, 42 (2019), 9008–9018.

[3] Kubečka, J., Besel, V., Kurtén, T., Myllys, N., and Vehkamäki, H. Configurational sampling of noncovalent (atmospheric) molecular clusters: sulfuric acid and guanidine. *J. Phys. Chem. A 123* (2019), 6022–6023.

[4] Myllys, N., Kubečka, J., Besel, V., Alfaouri, D., Olenius, T., Smith, J. N., and Passananti, M. Role of base strength, cluster structure and charge in sulfuric-acid-driven particle formation. *Atmos. Chem. Phys. 19*, 15 (2019), 9753–9768.

[5] Valiev, R. R., Hasan, G., Salo, V.-T., Kubečka, J., and Kurtén, T. Intersystem crossings drive atmospheric gas-phase dimer formation. *J. Phys. Chem. A 123*, 30 (2019), 6596–6604.

Installation

HOW TO INSTALL

```
git clone https://github.com/kubeckaj/JKCS2.1.git #clone JKCS2.1
cd JKCS2.1                                        #enter the directory
sh setup.sh                                       #run the set up (installation)
source ~/.bashrc                                  #JKCS can now be run anywhere
```

* 'sh setup.sh -r' can be used to overwrite an already existing user set up (*~/.JKCSusersetup.txt*)
** for users unfamiliar with BASH: *~/.bashrc* contains the paths to executable scripts. This allows the scrips to be used in any directory without having to define the path each time.

ADJUST THE USER SET UP

```
vim ~/.JKCSusersetup.txt                          #adjust user setup

# CHANGE THE FOLLOWING PATHS/MODULES
# ABCluster: rigidmoloptimizer path
# XTB:       XTB path
# PYTHON:    please set up how to use python2.0
#            (python 3.0 is just for GoodVibes in JKCS4)
# Gaussian:  G16 path and modules
# Orca:      Orca path and modules

sh test.sh                                        #test that all your paths
                                                  #are set correctly
```

* H. Vehkamäki group: Puhti users do not have to adjust anything

At the end of every section of this manual where a JKCS script is explained, the programs needed to run that script are specified (e.g. Python3.x is required just for **JKCS4_collect** when applying GoodVibes calculations).

After writing "JK" and pressing tabulator [TAB] in the command line, you should be able to see various JK & JKCS commands (see Figure 2).

```
USER: JK[TAB]
JKattachstr.py    JKCS1_prepare     JKCS5_filter      JKg16.sh          JKname            JKremovefiles     JKsubg16
JKavg             JKCS2_explore     JKCS6_monster     JKgaussstat       JKorca            JKrevibPM7        JKxyz2com
JKcheck           JKCS2_runABC      JKfor             JKlog2com         JKorcaDLPNO       JKsacct           JKxyz2inp
JKchmod           JKCS3_run         JKformation       JKlog2xyz         JKqdelall         JKsend
JKCS0_copy        JKCS4_collect     JKfreeslots.py    JKmovetoorigin.py JKqmuch           JKsend8
```

Figure 2: Auto-filling options for JK & JKCS commands.

First test

After installing JKCS, you can now perform a short configurational sampling test to see if everything is correctly set up and to get a hang of the workflow. Since these test calculations should be very fast, the command argument "-loc" is used in several cases to run jobs on the local system.

```
cd $WRKDIR                  #go to your working directory
mkdir TEST_JKCS             #create a new folder
cd TEST_JKCS                #and enter it
JKCS0_copy --help           #This fails if installation wasn't successful or
                            #you didn't source ~/.bashrc. Shows help
JKCS0_copy SA A             #creates input.txt for sulf. acid and ammonia
ls                          #list copied files
vim input.txt               #make changes to the input file //or nano,emacs
JKCS1_prepare --help        #shows help
JKCS1_prepare               #prepares folders and some files based on input.txt
ls                          #lists files and folders
JKCS2_explore --help        #shows help
JKCS2_explore -gen 10 -lm 5 -pop 10 -loc
                            #this fails if you didn't set up ABCluster
                            #runs 10-bee colony for 10 generations and save 5
                            #best structures (each subfolder is entered).
JKcheck                     #check that all calculations were finished
#optional: you can perform [JKCS4_collect ABC -loc] to collect ABC results
JKCS3_run --help            #shows help
JKCS3_run -loc              #This fails if you didn't set up XTB path
JKCS4_collect XTB -loc      #This fails if you didn't setup python2.x properly
                            #Collects semi-empirically(XTB) optimized xyz
cd SYS_1SA_1A               #Enter folder with cluster 1SA_1SA
cat resultsXTB.dat          #see results [structure | gyration rad. | energy]
molden movieXTB.xyz         #visualize molecules
```

If everything worked, you can continue. Otherwise, try to adjust the rest of file *~/.JKCSusersetup.txt*: QUEUING, QUANTUM CHEMISTRY PROGRAMS, *etc.* You can also contact Jakub Kubečka (jakub.kubecka@helsinki.fi) if problems persist.

The input.txt file

Configurational sampling with JKCS starts from the *input.txt* file. The *input.txt* file can be generated by using **JKCS0_copy** (see next chapter). This file contains the most import definitions for sampling the desired clusters in the current directory. The *input.txt* file is divided into four parts:

- Supercomputer parameters
- System charge and multiplicity
- Composition
- Structure of building monomers

We will go over each of these parts. For illustration, you can look at Figure 3.

SUPERCOMPUTER PARAMETERS
In this section, the parameters for running JKCS on a supercomputer are defined. JKCS communicates with the 3rd-party programs ABCluster, XTB, Gaussian16 and ORCA. For each of these programs, the following parameters need to be specified:

- Maximum number of tasks that can run in parallel
- Number of CPUs
- Number of nodes
- The requested wall time
- The name of the supercomputer partition
- The memory per CPU

The -loc program is used for computing on a local computer. For almost all of them, you can use variables NoC and M to define the parameter as a function of 'Number Of Combinations' or 'Number of Molecules'.

When calling one of the JKCS scripts (e.g. **JKCS3_run**), the supercomputer parameters for running that script can also be specified as additional arguments to the script. We could, for instance, call 'JKCS3_run XTB -maxtasks 20 -mem 8g' to change the maximum number of tasks and memory per CPU from what is written in the *input.txt* file. These commands will be further explained in the 'Supercomputer commands' section of this manual.

The default table for supercomputer parameters can be changed in *~/.JKCSusersetup.txt*.

SYSTEM CHARGE AND MULTIPLICITY
This section consists of two parameters that need to be set: the total charge and total multiplicity of the cluster(s). The multiplicity is equal to the number of unpaired electrons plus one ($M = 2S+1$).

COMPOSITION

The composition defines the number of each monomer in the desired cluster. For each cluster, the composition is written as $n_1\_n_2\_n_3 \ldots \_n_z$, where n_i is the number of monomers of type i in the desired cluster. The order in which the monomers appear in this format should be the same as the order in which the monomers are listed in the 'structure of building monomers' part of the *input.txt* file (see below). When configurational sampling of multiple clusters with different compositions need to be done, each composition can be written on one line with a space between two separate compositions. For multiple clusters, some symbols can also be used to quickly define the clusters. Writing "1-3_(4,5)" would for instance be equivalent to "1_4 2_4 3_4 1_5 2_5 3_5".

STRUCTURE OF BUILDING MONOMERS

This section should list the name, path and charge of all available conformers and conjugate acids/bases for all the monomers in the desired cluster. All conformers and conjugate species of the same molecule should have the same name (e.g. SA for *cis*- and *trans*-sulfuric acid as well as for the bisulfate and sulfate ion). The order of the list should be the same as the order in which the composition was written.

EXAMPLE

Consider as an example that we would like to perform configurational sampling on a negative cluster containing two sulfuric acid molecules and one ammonia molecule. For this, we would change the total charge of the cluster to -1. The multiplicity would be left at 1. The composition would be given as 2_1. Lastly, we fill in the 'structure of building monomers' part. As we have filled in 2_1 in the composition, we first list all sulfuric acid conformers and conjugate bases and then we list the ammonia structure and conjugate acid. There are two sulfuric acid conformers to take into account: *cis*- and *trans*-sulfuric acid. Ammonia has only one conformation. We take multiple conformers of monomers into account because it is not certain that the lowest energy monomer conformer is also the preferred conformation inside the cluster. Sulfuric acid has two conjugate bases: bisulfate and sulfate. Ammonia has one conjugate acid: ammonium. We consider the conjugate acids and bases because internal acid-base reactions could occur between monomers in the cluster. The paths to all the different structures related to one monomer should be listed together.

The example of a cluster containing two sulfuric acid molecules and one ammonia molecule will be used throughout this manual.

CREATING THE INPUT.TXT FILE

JKCS contains optimized structures for the different conformers and conjugated acids/bases of some of the most prominent monomers in atmospheric clusters (such as sulfuric acid and ammonia). The *input.txt* file for a cluster containing these monomers can be created automatically by calling the **JKCS0_copy** script. The **JKCS0_copy** script and the available monomers will be detailed in the following section. When configurational sampling needs to be done on a cluster that contains monomers not included with JKCS, then the input.txt file can be created from scratch. It is of course also possible to let **JKCS0_copy** create an *input.txt* file for a random system and subsequently change the necessary parameters to fit the desired system.

It is important to note that even when creating the *input.txt* file through **JKCS0_copy**, it is likely that some parameters, like the composition, still need to be altered.

```
#####################################################
## SUPERCOMPUTER PARAMETERS                        ##
## Number of Combinations - NoC                    ##
#####################################################
## MAXTASKS CPU NODES REQ.TIME  PARTITION MEMPERCPU ##
=====================================================
ABC  NoC     1    1   72:00:00   serial     4000
XTB  NoC     1    1   72:00:00   serial     4000
G16  100     8    1   72:00:00   serial     4000
ORCA 100     8    1   330:00:00  longrun    4000
-loc 1       1    1   -          -          -
=====================================================

####################################
## SYSTEM CHARGE AND MULTIPLICITY  ##
####################################
TotalCharge        0
TotalMultiplicity  1

#####################################################################
## COMPOSITION:                                                    ##
## e.g.: 1_1_2 1_2_1 1_3                                           ##
## e.g.: 1_3-6             = 1_3 1_4 1_5 1_6                       ##
## e.g.: (1,3,5)_1         = 1_1 3_5 5_1                           ##
## e.g.: (2,4)_1-3_1       = 2_1_1 4_1_1 2_2_1 4_2_1 2_3_1 4_3_1 ##
## e.g.: 1_1_F2            = 1_1_0-2 #protons# to fulfill charge ##
#####################################################################
Composition     1_1  2_2

####################################
## STRUCTURES OF BUILDING MONOMERS: ##
####################################
# name | q | path
  SA     0   /wrk/kubeckaj/DONOTREMOVE/Apps/JKCS2.0/JKCSx/../TOOLS/STRUCTURES/ABC/h2so4.xyz
  SA     0   /wrk/kubeckaj/DONOTREMOVE/Apps/JKCS2.0/JKCSx/../TOOLS/STRUCTURES/JACOB/h2so4_cis.xyz
  SA    -1   /wrk/kubeckaj/DONOTREMOVE/Apps/JKCS2.0/JKCSx/../TOOLS/STRUCTURES/ABC/hso4.xyz
  SA    -2   /wrk/kubeckaj/DONOTREMOVE/Apps/JKCS2.0/JKCSx/../TOOLS/STRUCTURES/ABC/so4.xyz
  A      0   /wrk/kubeckaj/DONOTREMOVE/Apps/JKCS2.0/JKCSx/../TOOLS/STRUCTURES/ABC/nh3.xyz
  A      1   /wrk/kubeckaj/DONOTREMOVE/Apps/JKCS2.0/JKCSx/../TOOLS/STRUCTURES/ABC/nh4.xyz
```

Figure 3: The *input.txt* file in its full beauty.

0 - JKCS0_copy

JKCS0_copy automatically creates an *input.txt* file based on the monomers given as arguments. JKCS contains optimized structures for the different conformers and conjugated acids/bases of some of the most prominent monomers in atmospheric clusters (such as sulfuric acid, ammonia, nitric acid or water). When calling **JKCS0_copy** with the name of some monomers as arguments, an *input.txt* file will be created containing the paths to the optimized structures of the conformers and conjugated acids/bases of all the requested monomers in the "structure of building monomers" part of the file.

It is once again important to note that even when creating the input.txt file through **JKCS0_copy**, it is likely that some parameters, like the composition, still need to be altered. **JKCS0_copy** will write two example compositions for the chosen monomers. These of course do not necessarily have to coincide with the desired cluster composition. Therefore, always check the *input.txt* file before proceeding.

COMMAND

```
JKCS0_copy [OPTION(s)] [STRUCTURE(s)]
```

OPTIONS

```
-help ........ print this help
-all ......... link all available structures
```

STRUCTURES

```
W ............ water (H2O,OH-,H+)     CO2 .......... carbon dioxide
AQ ........... water (H2O)            CH4 .......... methane
                                      Ar ........... argone
HNO3 ......... nitric acid            Ne ........... neone
SA ........... sulphuric acid         He ........... helium
MSA .......... methanesulfonic acid
                                      H ............ proton (+)
A,AM ......... ammonia                Na ........... sodium (+)
GD ........... guanidine              Cl ........... chloride (-)
DMA .......... dimethylammine
TMA .......... trimethylammine       HIO3 ......... iodic acid
urea ......... urea                  HIO2 ......... iodous acid
                                      I2O5 ......... iodine pentoxide
```

EXAMPLES

```
JKCS0_copy SA A  #creates input file containing links to SA and A structures
```

Consider again the example of a negatively charged cluster containing 2 sulfuric acid molecules and 1 ammonia molecule. By writing 'JKCS0_copy SA A', an *input.txt* file is created. This *input.txt* file will already contain the information of cis- and trans-sulfuric acid, bisulfate, sulfate, ammonia and ammonium in the 'structure of building monomers' section. The total charge and composition will however not be the same as our desired cluster. We will thus have to change the total charge to "-1" and the composition to "2_1".

If we would have called 'JKCS0_copy A SA', ammonia and ammonium would first be mentioned in the 'structure of building monomers' section followed by the sulfuric acid monomers and conjugated bases. In this case, we should write the composition as "1_2" to obtain our desired cluster.

REQUIRED

```
bash/sh
```

1 - JKCS1_prepare

The **JKCS1_prepare** script uses the information in the *input.txt* file to create *SYS_{system}* subfolders in your working directory. Each subfolder *SYS_{system}* corresponds to a specific cluster composition of the *input.txt* file. **JKCS1_prepare** also creates a file *parameters.txt* inside each of these subfolders. The *parameters.txt* file is similar to the *input.txt* file and is used further for additional calculations in each subfolder. **JKCS1_prepare** has however added all possible combinations of conformers and conjugated acids/bases of monomers, that fulfil the composition and total charge as defined in the *input.txt* file, to the end of the *parameters.txt* file. These combinations will be used in the configurational sampling. **JKCS1_prepare** does not require any arguments.

Each subsequent script (for instance **JKCS2_explore** or **JKCS3_run**) can either be called from the work directory or from one of the *SYS_{system}* subfolders. When called from the work directory, the script is applied to all subfolders. When called from a subfolder, it is only applied to that specific subfolder.

COMMAND

```
JKCS1_prepare [OPTION(s)]
```

OPTIONS

```
-help ................. print this help
```

ADVANCED OPTIONS

```
-s,-sample "X" ........ in the case of many combinations, picks just X
                        random combinations (useful when molecules with
                        a lot of isomers are used)
-o,-overwrite ......... overwrite parameters.txt also in existing folders
```

* The variable X could be a number (*e.g.*, 100) or a function of number of Molecules (M) or Number Of Combinations (NoC) (*e.g.*, "20*M", "NoC/2", "2*NoC/M")

EXAMPLE

Calling **JKCS1_prepare** for our illustrative negatively charged cluster of 2 sulfuric acid molecules and 1 ammonium molecule, one folder SYS_2SA_1A would be created. This folder would contain a *parameters.txt* file where 5 possible cluster combinations are written:

- 0_0_2_0_0_1 ($2HSO_4^-$_NH_4^+)
- 0_1_0_1_0_1 (*cis*-H_2SO_4_SO_4^{2-}_ NH_4^+)
- 0_1_1_0_1_0 (*cis*-H_2SO_4_ HSO_4^-_NH3)
- 1_0_0_1_0_1 (*trans*-H_2SO_4_ SO_4^{2-}_ NH_4^+)
- 1_0_1_0_1_0 (*trans*-H_2SO_4_ HSO_4^-_NH_3)

You can easily verify that these clusters all have total charge -1 and that these are all the combinations that can be made for this specific composition and total charge.

REQUIRED

```
bash/sh
python = python2.x
```

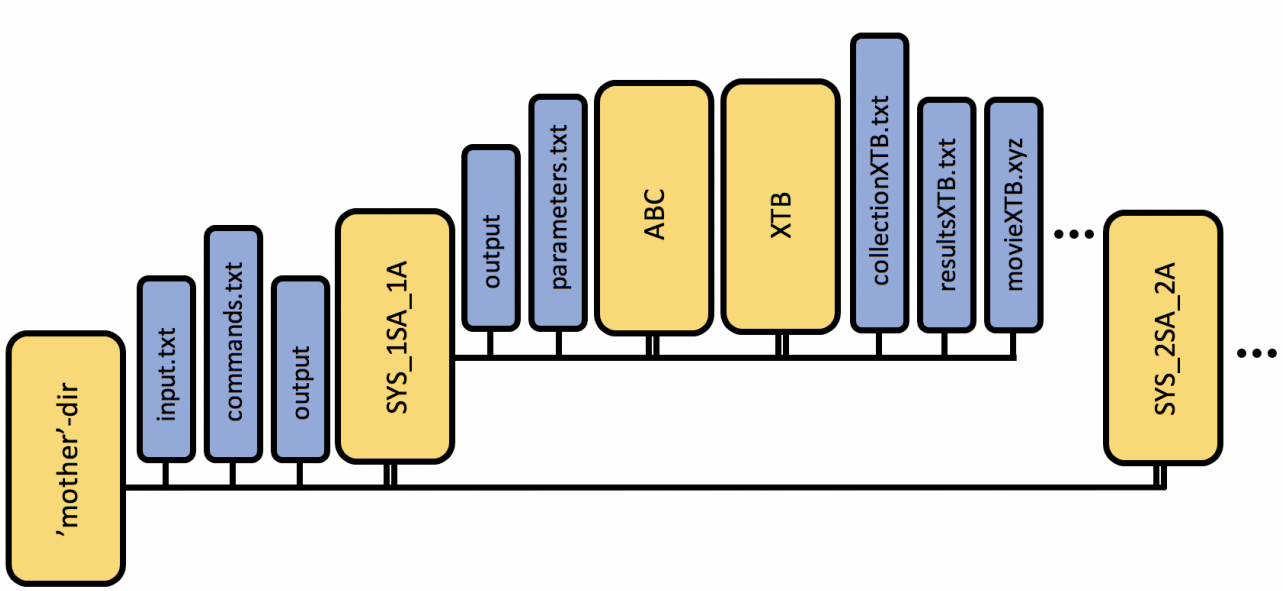


Figure 1.1: Scheme of folders and files formed during the configurational sampling.

Supercomputer cluster

When working on a (super)computer cluster, all parameters for submitting jobs are taken from the file *parameters.txt*. This file is created by **JKCS1_prepare** based on the user input file *input.txt*. When calling any additional script of JKCS, it is possible to overwrite the supercomputer parameters in *parameters.txt* by adding the following arguments to the specific command. The order of the arguments is not important. It only makes sense to use these arguments when calling **JKCS2_explore**, **JKCS3_run** or **JKCS4_collect**. These scripts employ 3rd party programs that might demand considerable computational resources. (NOTE: JKCS3_run -greasy)

ARGUMENTS

```
-loc ..................... run the calculation on local computer
-tasks,-maxtasks NUMBER .. max. number of tasks to be submitted (per subfolder)
-jpt NUMBER .............. jobs per task (e.g. 3 task submitted each with 6 jobs
                           = 18 jobs calculated overall) [def=1]
-cpu NUMBER .............. amount of CPU(s) used per one job
-N,-nodes NUMBER ......... amount of nodes to be utilized [NOT TESTED]
-time TIME_FORMAT_NUMBER . requested walltime [e.g., 72:00:00]
-par,-partition NAME ..... partition name [e.g., short, hugemem, longrun ...]
-mem,-memory MEMORY ...... size of memory allocated per CPU [e.g., 4000mb]
```

EXAMPLES

```
JKCS2_explore -pop 2000 -gen 300 -lm 4000 -tasks 4 -mem 6000mb
                              #run JKCS2_explore for a maximum of 4 parallel
                              #tasks and with a memory of 6000mb. The number of
                              #CPUs, the partition and the wall time are still
                              #taken from parameters.txt.

JKCS3_run -program XTB -of ABC -mem 1GB -cpu 2 -par test -time 00:20:00
                              #run JKCS3_run with 2 CPU and a memory of 1 gb
                              #per CPU for 20 minutes in the test partition.
                              #The maximum number of tasks and number of nodes
                              #is still taken from parameters.txt.

JKCS3_run -maxtasks 1 -jpt 8  #run JKCS3_run with a maximum of one task which
                              #computes 8 jobs. All other supercomputer para-
                              #meters are taken from parameters.txt.

JKCS4_collect DFT -loc        #run JKCS4_collect on the local computer
```

REQUIRED

```
bash/sh
sbatch
```

2 - JKCS2_explore

The **JKCS2_explore** script employs the ABCluster program [2, 3] to search for the global and local minima of the different cluster combinations in the selected subfolder(s). The ABCluster program implements the Artificial Bee Colony (ABC) algorithm. This algorithm was first proposed by Karaboga [1] in 2008.

For every cluster combination, the ABCluster program creates a specified number of guess structures (pop = population) by adding the monomers of that cluster combination together at random orientations with respect to each other. Each guess structure is optimized by using information of all the other guess structures. This optimization will be run for a specified number of steps (gen = generations). If the potential energy of a specific guess structure does not change for more than a specified number of steps (s), the structure is replaced by a new guess structure. This ensures that the algorithm does not get stuck in any local minimum for too long. At the end of all the optimization steps, you can decide how many optimized minima are saved as output (lm = local minima).

COMMAND

```
JKCS2_explore [OPTION(s)] [STRUCTURE(s)]
```

DEFAULT COMMAND

```
JKCS2_explore =
  JKCS2_explore -program ABC -lm 300*M/NoC -gen 100 -sc 4 -pop 300*M
```

OPTIONS

```
-help ............. print this help and exit
-l,-lm "X" ........ X local minima will saved                  [def=300*M/NoC]
-g,-gen "X" ....... number of ABC generations                  [def=100]
-s,-sc "X" ........ steps before replacing unchanged structure [def=4]
-pop,-i,-init "X" . number of initial guesses                  [def=300*M]
```

ADVANCED OPTIONS

```
OTHERS: -box [def=7+M]
OTHERS: -wtb,-walltimebased {1000s,3h,1d ...}      [TESTING]
OTHERS: -wtc,-walltimecontrolled {1000s,3h,1d ...} [TESTING]
```

Some variables are by default defined as functions of the 'number of Molecules' (M) and the 'Number of Combinations' (NoC). This is particularly useful when working with several clusters. Smaller cluster do not need such extensive exploration as larger clusters. M and NoC allow for the scaling of the exploration according to the cluster size and complexity. You can use these symbols as well to define arguments.

SYMBOLS

```
M .... number of molecules
NoC .. number of combinations to form cluster
* and / can be used in combination with M or NoC to multiply or divide
```

It is important to note that when a lot of conformers are taken into account for a certain monomer, NoC can become very large. As a result, M/NoC becomes very small. Therefore, always be mindful of the result that using M and NoC might have for the exploration of all your studied systems. When dealing with large NoC, we believe that better performance of configurational sampling is reached when doing a small exploration on all combinations, rather than an exhaustive exploration of only a few selected combinations.

Some command arguments are not rigorously tested yet but can already be used. These are -wtb and -wtc:
WALL-TIME-BASED: "-wtb XYh" sets the number of generation equal to 100 and the population in such a way that the overall calculation takes XYh time to be finished by 1 CPU.
WALL-TIME-CONTROLLED: "-wtc XYh" works like "-wtb XYh" except that it overwrites population and generation if they are higher, i.e. the calculation would take more than XYh.

EXAMPLES

```
WRONG exploration (BAD = DO NOT USE):
        JKCS2_explore -pop 100 -lm 100 -gen 1     #too short (gen>=100)
        JKCS2_explore -pop 1000 -gen 100 -lm 1    #few saved structures
        JKCS2_explore -pop 100 -gen 100 -lm 300   #too small population

Correct exploration:
        JKCS2_explore -pop 1000 -gen 100 -lm 3000
        JKCS2_explore -pop 1000*M -gen 100 -lm 4000/NoC
        JKCS2_explore -wtc 3h -pop 2000*M/NoC
```

For our 2SA_A cluster, we could use any of the correct explorations. 2SA_A has 3 molecules and 5 possible combinations. The second correct exploration would thus use a population of 3000 guess structures and will save 800 local minima.

REQUIRED [2, 3]

```
bash/sh
ABCluster: rigidoptimizer
```

CITATION

[1] Karaboga, D., and Basturk, B. On the performance of artificial bee colony (ABC) algorithm. *Appl. Soft Comput. 8* (2008), 687–697.

[2] Zhang, J., and Dolg, M. ABCluster: the artificial bee colony algorithm for cluster global optimization. *Phys. Chem. Chem. Phys. 17* (2015), 24173–24181.

[3] Zhang, J., and Dolg, M. Global optimization of rigid molecular clusters by the artificial bee colony algorithm. *Phys. Chem. Chem. Phys. 18* (2016), 3003–3010.

3 - JKCS3_run

The **JKCS3_run** script utilizes 3rd party programs to optimize clusters and evaluate their properties (*e.g.*, electronic energy or free energies). The script either uses the output structures in the folder of a previous calculation (for instance, XTB, ABC, DFT_opt) as input structures for the new calculation, or uses the files written in a *resultsXXX.dat*/*resultsXXX_FILTERED.dat* file as input. *resultsXXX.dat* and *resultXXX_FILTERED.dat* files are created by JKCS4_collect and JKCS5_filter respectively (see the next two sections of this manual). When both a *resultXXX.dat* and a *resultXXX_FILTERED.dat* file are available, **JKCS3_run** will give preference to the *resultXXX_FILTERED.dat* file.

At the moment of writing, we use the following calculation steps in our research group [4, 5]:

- Semi-empirical optimization → GFN2-xTB [3, 1]
- High quality quantum chemistry optimization and vibrational analysis → Gaussian 16 [2]

  Final level of theory: DFT - ωB97X-D/6-31++g**
- Electronic energy correction → Orca [6, 7]

  Final level of theory: DLPNO-CCSD(T) - aug-cc-pvtz

Of course, **JKCS3_run** can be used with other workflows as well.

If you want to perform a GFN2-xTB step after ABC exploration, you can just use the command 'JKCS3_run' (see DEFAULT COMMAND).

COMMAND

```
JKCS3_run [OPTION(s)]
```

DEFAULT COMMAND

```
JKCS3_run =
  JKCS3_run -program XTB -oldfolder ABC -newfolder XTB -method "-opt vtight"
```

OPTIONS

```
-help ................. print this help
-nf,-newfolder NAME ... name of the new (calc.) folder
-of,-oldfolder NAME ... name of the old folder. The structures in this folder
                        will be used as input for the new calculation
                        (not compatible with -rf)
-rf,-resultsfile NAME . name of the resultsNAME.dat file containing
                        structures to be used for applying the comp. method
                        (File resultsNAME_FILTERED.dat is used, if it exists!)
                        (not compatible with -of)
-m,-method "X" ........ method used by 3rd party quantum program
-p,-program NAME ...... computational program to be used (XTB, G16, Orca ...)
                        program_$program has to be set up in
                        ~/.JKCSusersetup.txt
```

ADVANCED OPTIONS

```
-bs,-add,-addbase X .. insert basis set for heavy atoms to the end of file
                       (only for Gaussian)
```

EXAMPLES

```
XTB:
 JKCS3_run
 JKCS3_run -p XTB -nf XTB_freq -rf XTB -m "-ohess -temp 298.15"
G16:
 JKCS3_run -p G16 -rf XTB -nf DFT_sp   -m "# HF 6-31+g*"
 JKCS3_run -p G16 -rf XTB -nf DFT_opt  -m "# wb97xd 6-31++g** opt=verytight"
 JKCS3_run -p G16 -rf DFT_opt -nf DFT_freq -m "# wb97xd 6-31++g** freq"
ORCA:
 JKCS3_run -p ORCA -rf XTB -nf OPT -m "! PBE0 def2-TZVP TIGHTSCF Opt D3BJ"
ADVANCED:
 JKCS3_run -p G16 -rf XTB -nf DFT -bc I -m "#wb97xd GEN Pseudo=Read Opt
          Int=UltraFine Freq MaxDisk=32GB" -mem 12GB -cpu 16
 JKCS3_run -p ORCA -rf DFT_freq -nf DLPNO -m "! DLPNO-CCSD(T) aug-cc-pvtz
          aug-cc-pvtz/C GRID4 nofinalgrid TightPNO TightSCF NOPOP NOPRINTMOS"
```

Consider the first G16 example: JKCS3_run -p G16 -rf XTB -nf DFT_sp -m "# HF 6-31+g*". This command will create a Gaussian 16 calculation (-p G16). The input structures for the calculation are to be found in the *resultsXTB.dat* or *resultsXTB_FILTERED.dat* files (-rf XTB). **JKCS3_run** will create a new folder *DFT_sp* and store the input and output file of the current calculation in there (-nf DFT_sp). With Gaussian 16, a Hartree Fock single point energy calculation will be performed using the 6-31+g* basis set (-m "# HF 6-31+g*").

REQUIRED

```
bash/sh
3rd party program (XTB,Orca,Gaussian etc.)
```

CITATION

[1] Bannwarth, C., Ehlert, S., and Grimme, S. GFN2-xTBâĂŤAn accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions. *J. Chem. Theory Comput. 15* (2019), 1652–1671.

[2] Frisch, M. J., Trucks, G. W., Schlegel, H. B., Scuseria, G. E., Robb, M. A., Cheeseman, J. R., Scalmani, G., Barone, V., Petersson, G. A., Nakatsuji, H., Li, X., Caricato, M., Marenich, A. V., Bloino, J., Janesko, B. G., Gomperts, R., Mennucci, B., Hratchian, H. P., Ortiz, J. V., Izmaylov, A. F., Sonnenberg, J. L., Williams-Young, D., Ding, F., Lipparini, F., Egidi, F., Goings, J., Peng, B., Petrone, A., Henderson, T., Ranasinghe, D., Zakrzewski, V. G., Gao, J., Rega, N., Zheng, G., Liang, W., Hada, M., Ehara, M., Toyota, K., Fukuda, R., Hasegawa, J., Ishida, M., Nakajima, T., Honda, Y., Kitao, O., Nakai, H., Vreven, T., Throssell, K., Montgomery, Jr., J. A., Peralta, J. E., Ogliaro, F., Bearpark, M. J., Heyd, J. J., Brothers, E. N., Kudin, K. N., Staroverov, V. N., Keith, T. A., Kobayashi, R., Normand, J., Raghavachari, K., Rendell, A. P., Burant, J. C., Iyengar, S. S., Tomasi, J., Cossi, M., Millam, J. M., Klene, M., Adamo, C., Cammi, R., Ochterski, J. W., Martin, R. L., Morokuma, K., Farkas, O., Foresman, J. B., and Fox, D. J. Gaussian 16 Revision A.03, 2016. Gaussian Inc. Wallingford CT.

[3] Grimme, S., Bannwarth, C., and Shuskov, P. A robust and accurate tight-binding quantum chemical method for structures, vibrational frequencies, and noncovalent interactions of large molecular systems parametrized for all spd-block elements (z = 1-86). *J. Chem. Theory Comput. 13* (2017), 1989–2009.

[4] Kubečka, J., Besel, V., Kurtén, T., Myllys, N., and Vehkamäki, H. Configurational sampling of noncovalent (atmospheric) molecular clusters: sulfuric acid and guanidine. *J. Phys. Chem. A 123* (2019), 6022–6023.

[5] Myllys, N., Elm, J., Halonen, R., Kurtén, T., and Vehkamäki, H. Coupled cluster evaluation of the stability of atmospheric acid–base clusters with up to 10 molecules. *J. Phys. Chem. A 120* (2016), 621–630.

[6] Neese, F. The orca program system. *Wiley Interdiscip. Rev.: Comput. Mol. Sci. 2* (2012), 73–78.

[7] Neese, F. Software update: The ORCA program system, version 4.0. *Wiley Interdiscip. Rev.: Comput. Mol. Sci. 8* (2018), e1327.

JKgaussstat

The **JKgaussstat** script can be used on the output files created by **JKCS3_run** to print parameters of interest, such as the electronic energy, entropy, Gibbs Free energy, etcetera. The script is mainly utilized for Gaussian output. It can however also be used to collect certain parameters from XTB and ORCA output files.

By default, **JKgaussstat** prints the requested parameters directly in the command line. The user can of course also direct the **JKgaussstat** output into a file. The **JKCS4_collect** script employs **JKgaussstat** to collect parameters from all *SYS_{system}* folders and stores the output in a *results_XXX.dat* file (see the next section). In most workflows, users will work only with **JKCS4_collect**, instead of using **JKgaussstat** directly.

After using **JKgaussstat**, it is good practice to closely check what has actually been collected for several reasons: you might be using an incompatible version of Gaussian, you might have used **JKgaussstat** wrongly, or there could be an error in your script.

COMMAND

```
JKgaussstat [OPTION(s)] [FILES]
```

OPTIONS

```
-help ............ print this help and exit

The following options allow for the corresponding parameter to be printed. For
each option, the type of output file for which the option can be applied, is
specified (G=G16, X=XTB, O=ORCA, C=XYZ coord.).

-name ............ name of the file - (GXOC)
-p,-pwdname ...... full file path - (GXOC)
-b, -basename .... basename of the file - (GXOC)
-el, -elen ....... EL = (last) electronic energy (SCF Done) - (GXO )
-zpe, -ZPE ....... ZPE = zero-point energy (correction) - (GXO )
-g, -gibbs ....... G = Gibbs Free Energy (293.15 K) - (GXO )
-gh, -gibbsh ..... Gh = Gibss Free Energy (293.15 K) (GoodVibes) - (G   ) *
-gc, -gcorr ...... dGcorr = Gibbs Free Energy (293.15 K) correction - (GXO )
-gch, -gcorrh .... dGcorrh = Gibbs Free Energy (293.15 K) correction
                   (GoodVibes) - (GXO ) *
-dlpno ........... DLPNO = electronic energy from .out (ORCA output) - (GXOC)
-GD .............. DLPNO-G = DLPNO + dGcorr - (GXO ) **
-GDh ............. DLPNO-Gh = DLPNO + dGcorrh (GoodVibes) - (G   ) * **
-h, -enthalpy .... H = enthalpy energy - (GXO )
-HD, hdlpno ...... DLPNO-H = DLPNO - EL + H - (GXO )
```

```
-s, entropy ..... S = total entropy (hartree/K units) - (GXO )
-sh, -entropyh ... Sh = total entropy (hartree/K units) (GoodVibes) - (G   ) *
-Us, -Uentropy ... S = total entropy (cal/(mol.K) units) - (GXO )
-Ush, -Uentropyh . Sh = total entropy (cal/(mol.K) units) (GoodVibes) - (G   ) *
-rx, -ry, -rz .... rotational constant in 1 axis - (G   )
-r2 ............. calculate squared rotational constant - (G   )
-dx, -dy, -dz .... dipole moment in 1 axis (dy=0,dx=0) [Debye] - (GXO )
-dip, -d, -dtot .. total dipole moment = dz [Debye] - (GXO )
-pol ............ total polarizability [Angstrom^3] - (GX  )
-T, -temp "X" .... assuming temperature X in Kelvin - (G   )
-v "X" ........... anharmonicity scaling factor - (G   )
-t, -time ........ (total/last) time in minutes - (GXO )
-ht, -htime ...... (total/last) time in hours - (GXO )
-err ............ do not print for missing "Normal termination" - (G   )
```

* The commands "-gh", "-gch", "-GDh", "-sh" and "-Ush" make use of the GoodVibes program [1]. Goodvibes allows for the use of quasi-harmonic approximation corrections to thermochemical data from frequency calculations. In order to use these specific commands, the path of GoodVibes has to be set up properly in ~/.JKCSusersetup.txt.
** When using "-GD" and "-GDh", the corresponding ORCA DLPNO output should be located in the folder from which **JKgaussstat** is called.

EXAMPLES

```
JKgaussstat -p -b -rg -el -d 21_3_0.log
JKgaussstat -p -gh *.log > result.txt   # saves parameters to result.txt
JKgaussstat -gh *.log
JKgaussstat -GDh 21_11_15.log           # the DLPNO ORCA output 21_11_15.out
                                        # is located in the same folder
```

REQUIRED

```
bash/sh
python2.x (for Rg calculation)
python3.x (just for GoodVibes)
GoodVibes (for some specific variables: -gh ...)
+ chosen 3rd party quantum chemistry programs
```

CITATION

[1] Funes-Ardois, I., and Paton, R. Goodvibes: Goodvibes v1.0.1. *DOI: http://dx.doi.org/10.5281/zenodo.60811* (2016).

4 - JKCS4_collect

After **JKCS2_explore** or **JKCS3_run** calculations are finished, the **JKCS4_collect** script can be used to collect and order the output structures and their properties (such as dipole moment and electronic energy). **JKCS4_collect** utilizes the **JKgaussstat** script on all the output files in a specified calculation folder to find properties (such as file name, gyration radius, energy, or dipole moment) of all the structures. These properties are subsequently written into a *collectionXXX.txt* file, where XXX is the name of the specified folder from which the data are collected. The script furthermore performs some smart uniqueness filtering on the structures. After the filtering, a *resultXXX.dat* file is created, where redundant structures have been removed. The structures in the *resultXXX.dat* file are sorted with respect to energy (either Gibbs free or electronic energy). Moreover, all XYZ structures are also printed to *movieXXX.xyz*. Users can then visualize all the structures by using: molden *movieXXX.xyz*. File FILTER.txt shows the history of all uniqueness/filter/selection=sampling filtering.

As **JKCS4_collect** uses the **JKgaussstat** script to collect the properties of all output files, the same options can be used to request specific properties to be retrieved and written in the *collectionXXX.txt* and *resultXXX.dat* files. **JKCS4_collect** will always print the paths of the XYZ file containing the coordinates of the optimized structure, the gyration radii and the electronic energies as the first three columns of the *collectionXXX.txt* and *resultXXX.dat* files. Columns with additional properties can be requested by adding the corresponding options.

COMMAND

```
JKCS4_collect [OPTION(s)] [FOLDER]
```

DEFAULT COMMAND

```
JKCS4_collect =
  JKCS4_collect XTB
```

OPTIONS

```
-help ............ print this help and exit

JKCS4_collect uses the same options as JKgaussstat
```

We recommend to use the following commands when collecting data:

- XTB: “JKCS4_collect XTB -dip”
- DFT_opt: “JKCS4_collect DFT_opt”
- DFT_freq: “JKCS4_collect DFT_freq -gibbsh”
- DLPNO: “JKCS4_collect DLPNO -pXYZ -dlpno”

EXAMPLES

```
JKCS4_collect XTB
JKCS4_collect DFT_HIGH_freq -gibbs -loc
JKCS4_collect XTB -p -b -rg -el -dip
JKCS4_collect DFT_HIGH -loc
```

REQUIRED

```
bash/sh
python2.x (for Rg calculation)
python3.x (just for GoodVibes)
GoodVibes (for some specific variables: -gh ...)
+ chosen 3rd party quantum chemistry programs
```

5 - JKCS5_filter

The **JKCS5_filter** script provides different methods to filter the large number of structures in the configurational sampling. As input, **JKCS5_filter** takes the *resultsXXX.dat* files created by **JKCS4_collect**. After the filtering has been applied, the remaining structures are written in a *resultsXXX_FILTERED.dat* file. There are three different operations that **JKCS5_filter** can perform:

- Uniqueness filtering
- Outlier filtering
- Sampling (or selection)

Uniqueness filtering removes structures that are too similar to other structures. When calling **JKCS4_collect**, a uniqueness filter is automatically applied and the results are stored in a *resultsXXX.dat* file. This filter assumes two structures with energy difference less than 0.001 hartree and gyration radius (Rg) difference less than 0.01 to be the same. Users can run the **JKCS5_filter** uniqueness filter to fine-tune these uniqueness thresholds. Otherwise, users do not have to bother with uniqueness.

Outlier filtering removes structures that have properties too far away from that of the current lowest energy structure. This operation takes as an argument the threshold for specific properties above which a structure is considered an outlier. This filter uses both relative and absolute thresholds for the el. energy. We could ask for a rel. el. energy threshold (*e.g.*, '-d 3.5'). Any structure that has an el. energy more than 3.5 kcal/mol higher than the lowest energy structure will be removed. Or we could define an abs. el. energy threshold '-en' (*e.g.*, '-en 100'), and structures with energy higher than -100 Hartree will be removed. For the gyration radius, only absolute thresholds can be applied (*e.g.*, '-rg 6') removes all structures with gyration radius higher than 6 Å.

Sampling (or selection) uniformly (covering PES uniformly) picks a specified number of structures from the *resultsXXX.dat* file after applying outlier filtering. This is useful when there are too many structures left for calculations on a higher level of theory to be applied with a reasonable computational cost. As sampling omits some minima, it might result in losing the global minimum structure, but due to the uniform sampling, the global minimum should not be far from one of the other selected minimum.

These three filter operations can also be combined. You can thus call **JKCS5_filter** to apply an uniqueness and outlier filter on a *resultsXXX.dat* file and then uniformly pick a sample of the remaining structures.

After the XTB step, we strongly recommend to also collect the dipole moment and perform filtering using all 3 dimensions: Rg, E, and dip. See examples on how to use filtering in the end of this chapter.

The JKCS program also allows for the use of machine learning for filtering. This feature is still in testing phase at the moment of writing.

COMMAND

```
JKCS5_filter resultsXXX.dat [OPTION(s)]
```

OPTIONS

```
-help ............. print this help and exit

UNIQUENESS FILTERING
-u1 "X" ... The uniqueness threshold of Rg (the property in column 1)
            is set to 1E-X Angstrom
-u2 "X" ... The uniqueness threshold of the electronic energy
            (property in column 2) is set to 1E-X hartree
-u3 "X" ... The uniqueness threshold of the property in column 3 of the
            resultsXXX.dat file is set to 1E-X. The property in this column
            depends on which property was requested when calling JKCS4_collect
            (for instance dipole or Gibbs free energy)

OUTLIER FILTERING
-d "X" .... Remove outliers with rel. el. energy higher than X kcal/mol.
            compared to the lowest energy structure.
-dm "X" ... Remove outliers with rel. el. energy more than X*M kcal/mol
            compared to the lowest energy structure.
-en "X" ... Filter out all files with el. energy higher than X a.u.
-rg "X" ... Remove outliers with gyration radius higher than X Angstrom.
-rgm "X" .. Remove outliers with gyration radius more than X*M Angstrom
            higher than the lowest energy structure.

SAMPLING
-s "X" .... Uniformly sample X structures from the resultsXXX.dat file
-c3 "X" ... Besides the el. energy and gyration radius, the property in
            column X is used as a 3rd coordinate in the uniform sampling.
```

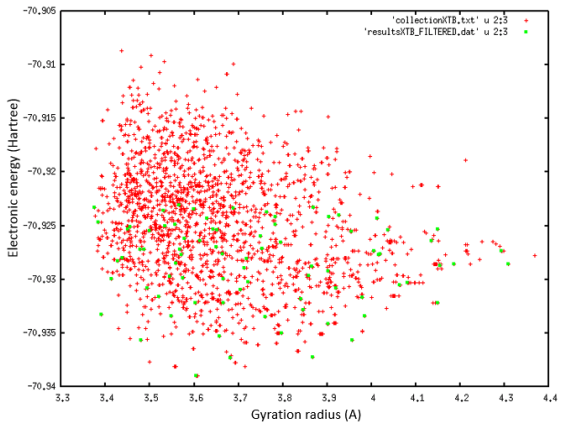
* M is the number of molecules in the cluster.

REQUIRED

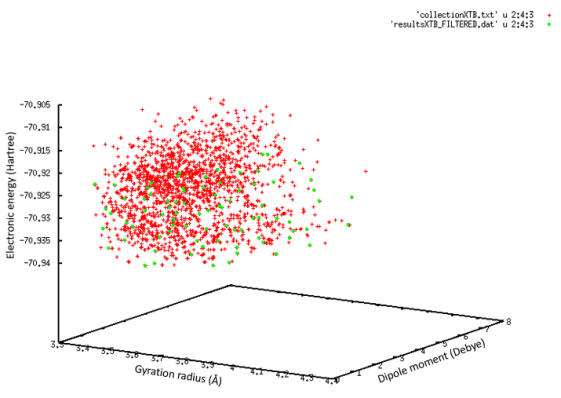
```
bash/sh
python2.x (for Rg, bonds or ncc calculation)
```

EXAMPLE IN 2-DIMENSIONS

```
Consider a situation where we have just finished an XTB calculation with
JKCS3_run on 2SA_A clusters. We now first have to collect the output data:
  JKCS4_collect XTB

This generates a collectionXTB.txt file and a resultsXTB.dat file, both with
3 columns: [filename path] [gyration radius (Rg)] [electronic energy (E)]

We can visualize the collected data using gnuplot:
  plot "collectionXTB.txt" u 2:3

Now we can apply JKCS5_filter.
  JKCS5_filter resultsXTB.dat -u1 2 -u2 4 -d 2.5 -s 100

The uniqueness filter '-u1 2 -u2 4' looks for structures with Rg difference
less than 0.01 (1E-2) and electronic energy difference less than 0.0001 (1E-4)
and keeps only one unique structure of them. The outlier filter '-d 2.5'
removes structures with el. energy more than 2.5 kcal/mol higher than the
lowest energy structure. After these filters are applied, '-s 100' picks
uniformly 100 of the remaining structures and writes them in a
resultsXTB_FILTERED.dat file.

We can again visualize the results of the filtering using gnuplot:
  p "resultsXTB.dat" u 2:3, "resultsXTB_FILTERED.dat" u 2:3 pt 5 ps 2
  (green points indicate the sampled structures)

A new calculation on top of the sampled structures can be performed like:
  JKCS3_run -rf XTB -nf DFT_opt -m "# wb97xd 6-31++g** opt=verytight"

The command '-rf XTB' take the structures in the resultsXTB_FILTERED.dat file
instead of the resultsXTB.dat file if a resultsXTB_FILTERED.dat file exists.
```



(a) Example of 2D rough filtering and sampling.



(b) Example of 3D rough filtering and sampling.

Figure 5.1: Example of filtering and sampling using different number of components.

EXAMPLE IN 3-DIMENSIONS

```
Consider again a situation where we have just finished an XTB calculation with
JKCS3_run on 2SA_A clusters. This time, we collect the output data and ask to
include the dipole moment:
  JKCS4_collect XTB -dip

A collectionXTB.txt and resultsXTB.dat file are created, both with 4 columns:
  - Column 1 contains the filename and path
  - Column 2 contains the gyration radius (Rg)
  - Column 3 contains the electronic energy (E)
  - Column 4 contains the dipole moment (D)

We can visualize the collected data using gnuplot:
  gnuplot
  plot "collectionXTB.txt" u 2:3:4
  (zoom in by mouse might be required: set yrange [0,10])

Now we can apply JKCS5_filter.
  JKCS5_filter resultsXTB.dat -u1 2 -u3 1 -rgm 3 -el -45.38 -c3 4 -s 80

The uniqueness filter '-u1 2' looks for  structures with Rg difference less
than 0.01 (1E-2) and '-u3 1' looks for structures with dipole moment
difference less than 0.1 (1E-1) and keeps only one unique of them. In this
specific case, the XTB energy is not taken into account (or you can add also
-u2 3). The outlier filters '-rgm 3' and '-d 3' removes structures with
gyration radius higher than 9 Angstrom (2SA_A has 3 molecules: 3*3=9) and el.
energy higher than -45.38 Hartree. After these filters are applied, '-s 80'
uniformly picks 80 of the remaining structures and writes them in a
resultsXTB_FILTERED.dat file. By adding '-c3 4' the uniform sampling will also
take the property of column 4 (dipole moment) into account.

#Generally, we recommend the following command:
#  JKCS5_filter resultsXTB.dat -rgm 3 -dm 3 -sm 40
#however, there might be cases, where the user has to adjust these settings.

Again, we can visualize the results of the filtering using gnuplot:
  gnuplot
  splot "resultsXTB.dat" u 2:3:4, "resultsXTB_FILTERED.dat" u 2:3:4 pt 5 ps 2
  (zoom in might be required: set yrange [0,10])
  (green points indicate our selection)

If we want to perform a new calculation on top of the sampled structures, we
can call JKCS3_run:
  JKCS3_run -rf XTB -nf DFT_opt -m "# wb97xd 6-31++g** opt=verytight"

The command '-rf XTB' will take the structures in the resultsXTB_FILTERED.dat
file instead of the resultsXTB.dat file if a resultsXTB_FILTERED.dat file
exists.
```

Additional JK scripts

This section gives a brief description of some useful additional scripts included in the JKCS program. You can use "--help" to further see how the following scripts work.

JKname
JKCS uses a unique set of numbers as filename for each structure (e.g. 21_11_15.log). Once the configurational sampling is complete, it is often helpful to change the filename to reflect the composition of the cluster. **JKname** helps you to rename your files based on their composition.

JKformation
This script calculates the formation Gibbs free energies of your system. When multiple minima are present, an additional argument "XY" representing temperature can be used to calculate formation Gibbs free energy assuming a Boltzmann distribution. Typical usage:

- collect all .log or .out files to one folder
- rename your files: JKname [FILENAME(s)]
- calculate dG: JKgaussstat -b -GDh -temp 273.15 > energies.txt
- calculate ddG: JKformation energies 273.15

JKcheck
This script helps you to see if one or multiple JKCS subfolder are still running some calculations or if all calculations are finished.

JKsymmol
This script uses the SYMMOL program [1] to calculate the rotational symmetry of a structure given in a XYZ file. The symmetry threshold can be given as an argument.

JKfor
JKfor is just a 'for' loop which enters each "SYS_xxx" subfolder and performs its arguments (*e.g.*: JKfor head -n 3 resultsDFTfreq.dat). You can use multiple commands if you enclose the entire argument by quotes (*e.g.*: JKfor "head -n 3 resultsDFTfreq.dat; head -n 3 resultsDLPNO.dat").

JKlog2xyz, JKxyz2com and JKlog2com
These scripts helps to convert between XYZ files, Gaussian input and Gaussian output.

JKxyz2inp
This script can create ORCA input files based on XYZ files.

CITATION

[1] Pilati, T., and Forni, A. SYMMOL: A program to find the maximum symmetry group in an atom cluster, given a prefixed tolerance. *J. Appl. Cryst. 31* (1998), 503–504.

Tips and useful information

GLOBAL MINIMUM
There is no guarantee that the global minimum of the cluster is actually found through configurational sampling. It becomes more probable if we include all monomer conformers, use a large number of guess structures and optimize more structures at higher levels of theory, but absolute certainty that the global minimum is found will never be achieved.

'M' & 'NoC' SYMBOLS
These symbols can be used with many JKCS commands to make values dependent on the 'number of Molecules' or 'Number of Combinations'. When a lot of conformers are taken into account for a certain monomer, 'NoC' can become very large. Therefore, always be mindful of the result that using 'M' and 'NoC' might have for the exploration of all studied systems.
Example: JKCS2_explore -pop 1000*M -gen 100 -lm 4000/NoC

COLORS & SYMBOLS IN PRINTED OUTPUT
The colors and symbols in the printed output of JKCS commands can be turned of in 2 ways:
1. Change Qsymbol or Qcolours in ~/.JKCSusersetup.txt to "no"
2. COLORING TEXT: use -nocolors argument to have text without colors
   KISSING SYMBOL: use -nosymbol to remove the symbol in the beggin. of output

DETAIL OF OUTPUT
The amount of printed output can be adjusted by using command '-print NUM':
-print 0 .... basically just error messages
-print 1 .... [DEFAULT] traditional output
-print 2 .... enlarged output, all algorithm steps are commented
-print 3 .... very very detailed output

CALLING JKCS COMMANDS
Each JKCS command can be performed inside a specific subfolder 'SYS_{system}', or from your 'mother directory' where the subfolders are located. In this case the algorithm enters each directory and perform the command there.

If you wish to perform a command from your 'mother directory', but only for some specific subfolders, you can give the subfolder as an argument:
Examples: JKCS2_explore SYS_3SA SYS_4SA -gen 100
          JKCS2_explore SYS_1SA_1-5AM -pop 2000 -gen 150

JKCS3_run can be used from any location outside the JKCS directory as long as a resultsXXX.dat file is present that contains the path to the XYZ-files to use in the calculation.

```
JK_CHECK
JKcheck can be used to check how many calculations have been finished.

ORDER OF ARGUMENTS
The order of arguments does not matter except for the following commands:
JKgaussstat, JKCS4_collect

JKCS6_MONSTER
JKCS6_monster can run multiple JKCS commands in succession. The commands to
execute are read from a txt file. Each command should be put on a separate
line. At the moment, JKCS6_monster is still being tested and a more in depth
help is currently not available.
Example: JKCS6_MONSTER commands.txt
```