eOn Documentation

Henkelman Group

August 11, 2010

1 Getting the Code

The code is available to users with an account on theory.cm.utexas.edu. It can be checked out of the subversion repository with the following command:

```
svn checkout svn+ssh://username@theory.cm.utexas.edu/Groups/svn/eon
```

This will fetch a copy of the latest code to a local directory eon. The work flow for making changes, after editing the files, is to first see what files you have modified:

```
svn status
```

This will give you a one line per changed file output of what you have done since you checked out the code. It is often a good idea to run:

```
svn update
```

next to make sure that no other developers have commited changes since you checked out your copy of the code you should run:

```
svn update
```

to get the latest copy of the code. It is possible that some other developer modified the same file in the same places that you have. This means that there will be some conflicts to resolve. This is a rare thing to happen and you should read the online documentation to figure out how to handle it. Once you have seen what changes you have made with 'svn status' and ensured that you have the latest version of the code with 'svn update' you should commit your changes to the repository:

```
svn commit -m "a brief message describing your changes"
```

2 Introduction

3 Input Files

Three input files are required to run an akmc simulation. The config.ini file which sets the options for the server code. The parameters.dat file which is passed on to the client and the initial configuration of the chemical system to be modeled.

3.1 config.ini

3.1.1 aKMC

Option: temperature
Input: *300* (Kelvin)
Option: thermally_accessible_window
Input: *20* (kT)
Option: maximum_thermally_accessible_window
Input: *2*
Option: confidence
Input: *0.95*
Option: max_kmc_steps
Input: *100000*

3.1.2 Paths

Option: main_directory
Input: *.*
Option: searches_out
Input: *%(main_directory)s/searches/out/*
Option: searches_in
Input: *%(main_directory)s/searches/in/*
Option: states
Input: *%(main_directory)s/states/*
Option: results
Input: *%(main_directory)s*
Option: scratch
Input: *%(main_directory)s/searches/scratch/*
Option: kdb
Input: *%(main_directory)s/kdbscratch/*
Option: superbasins
Input: *%(main_directory)s/superbasins/*
Option: superbasin_recycling
Input: *%(main_directory)s/SB_recycling/*

3.1.3 Communicator

Option: type
Input: *local*, mpi, boinc, cluster, arc
Option: search_buffer_size
Input: *100*
Option: job_bundle_size
Input: *10*
Option: client_path
Input: *eonclient*
Option: number_of_CPUs
Input: *1*
Option: mpi_command
Input: *mpirun -np %(number_of_CPUs)s mpi_wrapper*

3.1.4 Displacement

Option: type
Input: *random*, undercoordinated, leastcoordinated
Option: radius
Input: *5.0* (Å)
Option: size
Input: *0.01* (Å)
Option: maximum_coordination
Input: *11*

3.1.5 Recycling

Option: use_recycling
Input: *False*, True (to become default True?)

Whether or not recycling is implemented to try to recycle saddles from reference states.
Option: save_suggestions
Input: *False*, True

Whether or not the saddle suggestions that the recycler makes are saved.
Option: displace_moved_only
Input: *False*, True

Whether or not to only center displacements around atoms which moved getting from the recycling reference state to the current state.
Option: move_distance
Input: *0.2* (Å)

The distance an atom must have moved from the reference state to the current state for it to be considered "moved."
Option: use_sb_recycling
Input: *False*, True

Whether or not to try to find corresponding states in a new superbasin when exiting a superbasin and recycle from these rather than the immediately previous state.

3.1.6 KDB

Option: use_kdb
Input: *False*, True
Option: wait
Input: *False*, True
Option: keep
Input: *False*, True

3.1.7 Superbasins

Option: use_superbasins
Input: *False*, True
Whether or not superbasining is implemented (mutually exclusive with use_askmc).
Option: scheme
Input: *transition_counting*, energy_level
Which sheme is used to define superbasins during the simulation.
Option: number_of_transitions
Input: *5*
If "scheme" is set to number_of_transitions, this is the number of times a barrier between two states must be seen before the states are "superbasined."
Option: energy_increment
Input: *0.1*
Each time the state is visited a separate "copy" of its energy level is raised by 0.1 eV. When the energy of the copy reaches that of the barrier between states, those states are "superbasined."
XXX? I'm not exactly sure how these "superbasin (novotny)" ones work, so please check over them.
Option: use_askmc
Input: *False*, True
Whether or not accelerated superbasin kmc is implemented (mutually exclusive with use_superbasins).
Option: askmc_confidence
Input: (0.0 - 1.0), *0.9*
The confidence for AS-KMC.
Option: askmc_barrier_raise_param
Input: *1.5*
The parameter which sets how much the barriers are raised during AS-KMC.
Option: askmc_high_barrier_def
Input: *2*
The parameter which defines how high a barrier must be to be considered "high" in AS-KMC.

Option: askmc_barrier_test_on
Input: *True*, False

Whether or not the test to ensure no low barriers are missed during AS-KMC will be implemented.

Option: askmc_connections_test_on
Input: *False*, True

Whether or not the test to ensure no internal low-barrier connections are missed during AS-KMC will be implemented.

3.1.8 Structure Comparison

Option: energy_difference
Input: *0.1* (eV)
Option: distance_difference
Input: *0.05* (eV)
Option: use_identical
Input: *False*, True

3.1.9 Debug

Option: keep_bad_saddles
Input: *False*, True
Option: keep_all_result_files
Input: *False*, True
Option: random_seed
Input: *random*, integer
Option: register_extra_results
Input: *False*, True
Option: list_search_results
Input: *True*, False
Option: use_mean_time
Input: *False*, True

3.2 parameters.dat

3.3 reactant.con

4 Output Files

5 Config Options

5.1 Recycling

5.1.1 use_recycling

Input: *False*, True (to become default True?)

Recycling is a process whereby the saddle searches designed to find available processes for a given state take suggestions from a reference state. This can lead to moderate computational speedup, because the suggested displacements for the saddle searches are often quite close to the actual saddle, and thus convergence to the saddle requires fewer force calls. To obtain the displacement / mode suggestions, the reference state (normally the previous state unless "Recycling → use_sb_recycling" is set to True) and the current state are analyzed such that each atom is classified as either having moved significantly or not moved significantly getting from the reference state to the current state. Then for each viable process found for the reference state, a corresponding process is recommended to the current state. This is done by stepping through each atom in the reference state, and if it is "unmoved" compared to the current state, then its position in the suggested saddle is taken as that of its position in the process saddle of the reference state. If it is "moved" compared to the current state, then whatever motion it took in the process saddle is applied to the current state. Recycling meta-data is stored in the directory of each state ("main directory"/states/"#"/recycling_info).

For example, consider the case of two, non-adjacent adatoms on a 100 surface, each of which has four available processes, moving up/left/down/right. If, for example, the left one shifts to the left, the system is in a new state, but if recycling is on, much information can be gleaned from the previous state. The recycler steps through those 8 known processes from the previous state, and in each of the four in which the right (unmoved) adatom moves, its location in the saddle will be suggested directly. In each of the four in which the left (moved) adatom moves, its motion in the process will be applied to the current state, so it should have be reasonably close in each case to the four movement directions along the surface. Thus, recycling can lower the computational cost of finding saddle points.

A few options are available with recycling:

5.1.2 save_suggestions

Input: *False*, True

If save_suggestions is turned on, a sub-directory in each state directory called saddle_suggestions is created and the coordinate files of the suggestions are saved there.

5.1.3 displace_moved_only

Input: *False*, True

If this option is set to True, then the displacements which are made after recycling suggestions have been completed are centered only around the atoms which moved getting from the reference state to the current state. If there is not a list of moved atoms available (for example, if the system is still at state 0), then the displacement defined under "Displacement → type" is used. This can significantly decrease the number of searches required per state, but assumes

that atoms which are not 'close' to moved atoms will only have processes which were available in the reference state.

5.1.4 move_distance

Input: 0.2 (Å)

This defines the distance an atom must have been displaced from the reference state to the current state for it to be considered "moved."

5.1.5 use_sb_recycling

Input: *False*, True

This option dictates whether "superbasin recycling" is turned on. When set to true, this monitors the development of superbasins, and makes the assumption that when a superbasin is exited from, it is possible that the next state may be in a similar superbasin. In that case, rather than always using the previous state as the reference state for recycling, this will try to find corresponding states in a new superbasin and recycle from a better reference state to achieve highest recycle success.

Perhaps the easiest example involves a trimer and a single adatom on a 100 surface in which the rotation of the trimer may be considered a superbasin and the movement of the adatom is the most likely 'escape' from the superbasin to a new state. However, the trimer is likely to again be in a rotating basin. Thus when a superbasin is exited from, this method determines what "corresponding states" are likely to look like by finding the process from each superbasin state which involves similar movement to the process which led to the superbasin exit. If no such process exists for any of the superbasin states, then normal recycling resumes. Otherwise, superbasin recycling provides (hopefully) more appropriate reference states to the states in the new superbasin as they are stepped to in akmc.

5.2 Superbasins

5.2.1 use_superbasins

Input: *False*, True

5.2.2 scheme

Input: *transition_counting*, energy_level

5.2.3 number_of_transitions

Input: *5*

5.2.4 energy_increment

Input: *0.1*

5.2.5 use_askmc

Input: *False*, True

The accelerated superbasin kmc (AS-KMC) method was proposed by Chatterjee and Voter [1] to accelerate exit from superbasins during a kmc simulation, sacrificing the accuracy of intra-superbasin dynamics, but maintaining the accuracy in superbasin exit time and direction. This can drastically increase the timescales achievable in kmc simulations.

The basic process of AS-KMC involves gradually raising process barriers found to be inside of a superbasin such that exiting from the basin gradually becomes more likely. The method is designed to raise all the barriers in the superbasin simultaneously. Once a particular barrier has been crossed a certain number of times, N_f (more on determining N_f shortly), a check is performed to determine whether or not the current state is part of a superbasin. This is called the Superbasin Criterion. In the Superbasin Criterion, a search is performed, originating at the current state and proceeding outward through all low-barrier processes to adjacent states, and then through all low-barrier processes from each of these states, etc. For each low-barrier process found, if the process has been followed fewer than N_f times, the Superbasin Criterion fails and no barriers are raised. Thus, in the outward-expanding search from the originating state, the search continues until either a low-barrier process has been seen fewer than N_f times (and the Criterion fails) or until all connected low-barrier processes have been found and have been crossed at least N_f times (the edges of the superbasin are then defined and the Criterion passes). If the Superbasin Criterion passes, *all* the low-barrier processes (each of which as been crossed N_f times) are raised.

Note, that this method may be contrasted to the other implemented method to accelerate superbasin exit, referred to in the config file as Superbasins → use_superbasins. This method involves determining that the current state is in a superbasin by one of a number of methods, then, given the states in the superbasin, calculating the exact rates corresponding to exiting from the superbasin. Thus, no intra-superbasin dynamics are preserved (even inaccurate dynamics), but it may be expected that less error might result from using the other method.

Several parameters dictate the functioning of the AS-KMC method. These parameters dictate (1) how much the barriers are raised each time the Superbasin Criterion passes, (2) what defines a "low-barrier" for use in the Superbasin Criterion, and (3) the approximate amount of error the user might expect in eventual superbasin exit direction and time compared to normal kmc simulation. Also, two of these parameters in combination determine N_f, the number of times each low-barrier process must be seen before barriers may be raised in a superbasin.

5.2.6 askmc_barrier_raise_param

Input: *1.5*

[1] THE JOURNAL OF CHEMICAL PHYSICS 132, 194101 2010

(1) α – When barriers are raised in the simulation, they are raised such that the new rate constant is equal to the previous rate constant divided by α. Chatterjee and Voter recommend a value of 2 for many systems; however, when the time scales from the system are not known and time scales may overlap, they recommend letting $1 \leq \alpha \leq \gamma^{1/2} \ll N_f$.

5.2.7 askmc_high_barrier_def

Input: *2*

(2) γ – When performing the Superbasin Criterion, a process is considered a "low-barrier" process only in relation to the original process that started the Superbasin Criterion (the process seen with number of sightings $\geq N_f$). If the process in question has a barrier less than $ln(\gamma) \cdot kT$ more than the original process, it is considered to be a low-barrier process. Chatterjee and Voter recommend a value of 2 for most systems.

5.2.8 askmc_confidence

Input: (0.0 - 1.0), *0.9*

(3) δ – δ is a basic measure of the expected error in superbasin exit direction and time. Given a value for δ, the value of N_f is calculated based on the following:

$$N_f \geq \frac{\alpha - 1}{\delta} \cdot ln(\frac{1}{\delta})$$

Therefore, as δ approaches 0, N_f approaches ∞ and the simulation will be carried out nearly like normal kmc because the Superbasin Criterion will be rarely checked and pass. Thus no superbasin acceleration will be achieved either. Thus the higher the value of delta, the greater the computational speedup and the higher possible error.

In this software, "confidence" is set by the user, rather than δ directly, to avoid confusion with "confidence" and "error". Thus to achieve a delta of 0.2, the user would enter a confidence of 0.8.

Finally, because these simulations are actually AKMC, two extra options are provided to ensure the accuracy of the Superbasin Criterion:

5.2.9 askmc_barrier_test_on

Input: *True*, False

First, because the implemented Superbasin Criterion actually only considers processes which have been passed over at least once, there is some chance that a low-barrier process in a superbasin might have not been visited at all while all other low-barrier processes have been visited at least N_f times. This is unlikely, but this test verifies that such has not happened, considering even processes which have not been visited when determining if the Superbasin Criterion has passed. This check should not add significant overhead.

5.2.10 askmc_connections_test_on

Input: *False*, True

Second, there is a method, connections test, to ensure that there are no processes which connect states in the defined superbasin which have not been visited yet and which have a low-barrier. This check is somewhat more computationally expensive than the previous because structure comparisons have to be made when finding product states of unvisited processes.