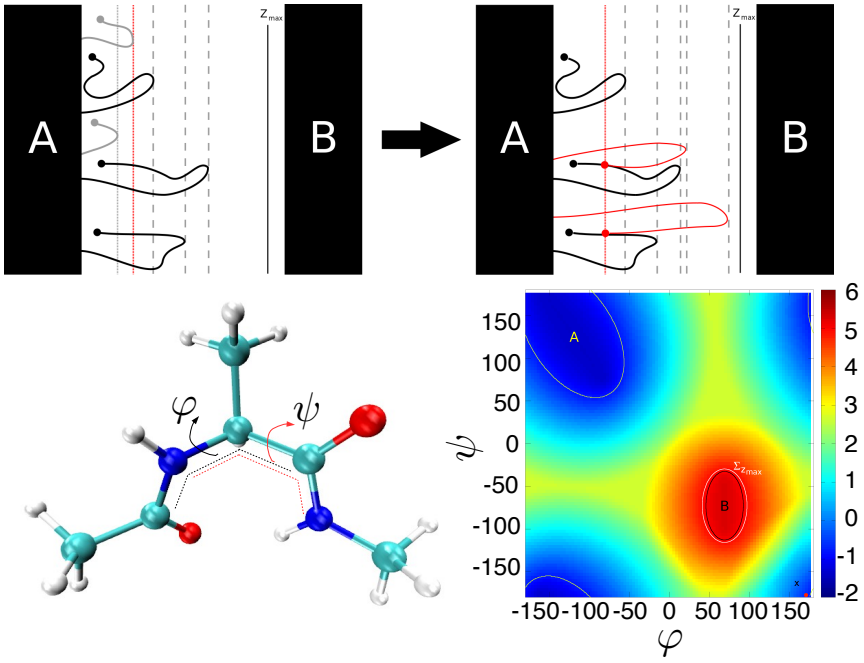
Centre d'Enseignement et de Recherche en Mathématiques et Calcul Scientifique
École des Ponts ParisTech

Centre National de la Recherche Scientifique
Laboratoire International Associé CNRS-UIUC
Université de Lorraine

University of Illinois at Urbana-Champaign
Beckman Institute for Advanced Science and Technology
Theoretical and Computational Biophysics Group

Adaptive Multilevel Splitting Method: Isomerization of the alanine dipeptide



Laura J. S. Lopes
Christopher G. Mayne
Christophe Chipot
Tony Lelièvre

January 18, 2018

Please visit www.ks.uiuc.edu/Training/Tutorials/ to get the latest version of this tutorial, to obtain more tutorials like this one, or to join the `tutorial-l@ks.uiuc.edu` mailing list for additional help.

Abstract

In this tutorial, we show how to apply the Adaptive Multilevel Splitting (AMS) method to the isomerization of the alanine dipeptide in vacuum. Section **1.** gives a description of the AMS algorithm and the proper way to set up AMS simulations, in the context of the NAMD program, for any system using the scripts provided with this document. An application of the method to the case example is showcased in section **2.**. More precisely, we show how to obtain the transition probability starting from one fixed point (Section **2.2.**), and the transition time (Section **2.3.**). Using the results obtained in these sections, a description of how to calculate the flux of reactive trajectories is given in Section **2.4.**. Results of the simulations for sections **2.2.** and **2.3.** are provided, so that the reader can straightforwardly go to Section **2.4.** if desired.

Contents

Completion of this tutorial requires:

- Files from `AMS_tutorial.zip` provided with this document
- NAMD version 2.10 or later
- Optional: Gnuplot

1. The Adaptive Multilevel Splitting method

The Adaptive Multilevel Splitting (AMS) method is a splitting method to sample reactive trajectories [1, 2, 3]. The goal here is to accelerate the transition between metastable states, which are regions of the phase space where the system tends to stay trapped. This method is particularly interesting because the positions of the intermediate interfaces, used to split reactive trajectories, are adapted on the fly, so they are not parameters of the algorithm. The AMS method was already efficiently applied to a large scale system to calculate unbinding time. [4]

Section **1.1.** presents the AMS algorithm as implemented in the Tcl script `ams.tcl`, provided with this document. In section **1.2.** we show how to set up AMS simulations for any system.

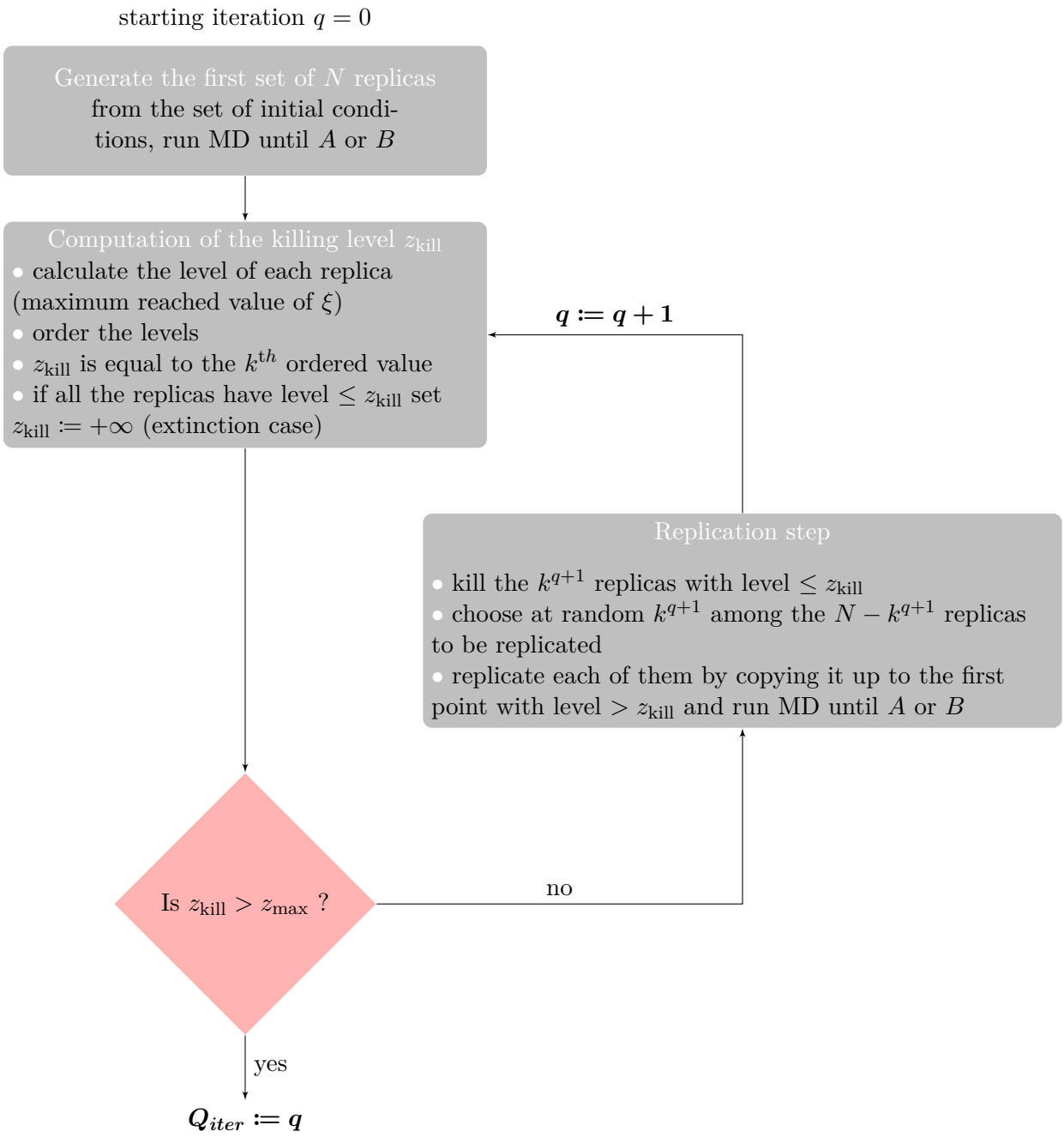
1.1. The AMS algorithm

Let us call A and B the source and target regions of interest, and assume that A is a metastable state. This means that starting from a point in the neighborhood of A, the trajectory is most likely to enter A before visiting B. The goal is to sample reaction trajectories that link A and B. In practice, these regions are defined using a set of internal variables of the system.

To compute the progress from A to B one needs to introduce a reaction coordinate ξ. Again, in practice ξ is a real-valued function of internal variables of the system. This function only needs to satisfy one condition: it is necessary that there exists a value of ξ that the system has to exceed to enter B when starting from A. This value of ξ is called $z_{\max}$. Note that the definitions of the zones A and B are independent of the reaction coordinate. Since ξ does not need to be continuous, the former condition can be enforced by making ξ equal to infinity on B. The condition is then satisfied with $z_{\max}$ equal to the maximum value of $\tilde{\xi}$ outside B. In practice, the easiest way is to make ξ equal to $z_{\max} + 1$ inside B.

The algorithm, as presented below, estimates the probability to observe a reaction trajectory, that is, coming from a set of initial conditions in a neighborhood of A, the probability to enter B before returning to A. We will denoted this estimator by p_{AMS}. This probability can be used to compute transition times and we will see how in Section **2.3.**.

The three numerical parameters of the algorithm are: (1) the reaction coordinate ξ, (2) the total number of replicas N, and (3) the minimum number k of replicas killed at each iteration. The algorithm starts at iteration $q = 0$ and follows the flowchart below (see also Figure 1 for a schematic representation).



Notice that at the end of iteration q, the estimation of the probability of reaching level z^q_{kill}, conditioned to the fact that level z^{q-1}_{kill} has been reached, (where by convention $z^{-1}_{\text{kill}} = -\infty$) is:

$$p^q = \frac{N - k^{q+1}}{N}. \tag{1}$$

Therefore, denoting r the number of replicas that reached B at the last iteration of the algorithm,

the estimator of the probability of transition is:

$$p_{\mathrm{AMS}} = \frac{r}{N} \prod_{q=1}^{Q_{iter}} p^{q-1} = \frac{r}{N} \prod_{q=1}^{Q_{iter}} \left(\frac{N-k^q}{N} \right), \tag{2}$$

where by convention $\prod_{q=1}^{0} = 1$. For example, if all the replicas in the initial set reached B, $r = N$ and, thus, $p_{\mathrm{AMS}} = 1$. In case of extinction $r = 0$, because no replica reached B, and thus $p_{\mathrm{AMS}} = 0$.

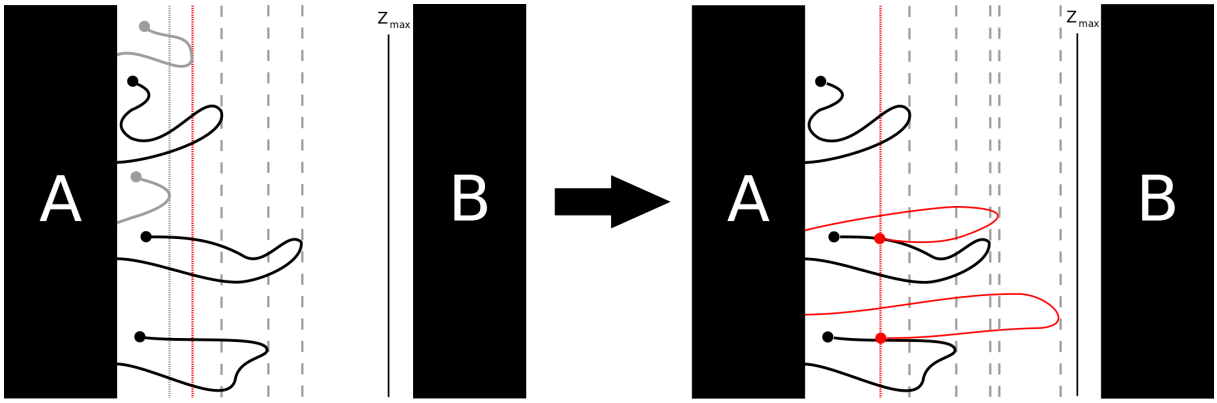


Figure 1: First AMS iteration with $N = 5$ and $k = 2$. Both lower level replicas (in gray) are killed. Two of the remaining replicas are randomly selected to be duplicated until level z^0_{kill} (red line) and then continued until they reach A (typically more likely) or B.

As the algorithm runs, all the points that can possibly be used in future replication steps must be recorded. In order to decrease the computational cost and the memory use, this is only done every $K_{\mathrm{AMS}} = \Delta t_{\mathrm{AMS}}/\Delta t$ timesteps. It is indeed useless to consider the positions of the trajectory at each simulation time step, as no significant change occurs over a 1 or 2 fs timescale.

1.2. Setting up AMS simulations

The AMS method is implemented in NAMD through a Tcl script sourced by the user in the configuration file, where the AMS functions are directly called. To run the algorithm the user must provide a set of files (see Section **1.2.1.**), and should have run the first set of replicas. A set of Tcl and bash scripts and simple C programs are provided to automate the process. As will be seen in the Section **1.2.2.**, the user will only need to provide one input file.

All the scripts and programs needed to run an AMS simulation can be found in the `smart` directory. The algorithm implementation is located in file `ams.tcl`. A script called `smart_parallel.sh` automates all the AMS runs, and it is the only script that the user will call directly. To utilize this script it is necessary to set a few variables.

1. Open script `smart/toall_path.sh` to edit it.
2. Set the variable `smart_path` as the path to all the smart files (i.e. to directory `smart`)
3. Set the variable `amsscript` with the `ams.tcl` script location.
4. (*) Provide the NAMD excutable file location through variable `namd`.
5. Close file `toall_path.sh`.
6. Open the terminal and type:
   `export toall_smart="/path/to/tutorial/files/smart/toall_smart.sh"`
   The export command not only defines a variable but makes its value visible for all the scripts that will be run in this same terminal session. To make it visible for all the sessions, just include this line into `/home/user/.bashrc` file.
7. Open file `common/namd.conf` to edit.
8. Set the variable `path` to the path to directory `common`.

(*) This should be a binary for a multicore build of namd (NOT build with CUDA).

1.2.1. The user files to provide

In addition to the basic NAMD files to run MD, it is necessary to provide a group of additional files to set up the AMS simulations. Some of these are Tcl scripts that should contains the definitions of a few procedures that will be called by the AMS Tcl script. If the reader is not familiar with Tcl language, procedure is the equivalent of function in Tcl. Nevertheless, it is not necessary to program in Tcl for this tutorial, as all these files are given in the `common` directory. The additional files in the `common` directory are:

- `dihedral_20.colv`: a Colvars [5] configuration file with the definition of the collective variables that will be used to calculate the regions A and B and the reaction coordinate ξ;
- `inzone.tcl`: a Tcl script with a procedure called `zone` that should return -1 if in region A, 1 if in B and 0 otherwise, using a set of the collective variables defined in the previous file;
- `coord.tcl`: a Tcl script with a procedure called `ams_measure` that has to return the value of the reaction coordinate ξ (also using the collective variables);

- `variables.tcl`: a Tcl script with a procedure called `variables` that returns a list of internal coordinates values used to visualize the reactive trajectories after the AMS run. In the case of alanine dipeptide this script only prints the collective variables defined in `dihedral_20.colv`. This scripts introduces flexibility to the analysis of the reactive trajectories, that can be made using different internal coordinates as the ones used to define A, B and ξ.
- `namd.conf`: a typical NAMD configuration file without any run step that will be the base to build all the NAMD configuration files for the AMS simulations.

9. Open file common/inzone.tcl and set the correct path to the executable file zones_CRI.
10. Do the same with file common/coord.tcl for the path to coord_CRI.

1.2.2. *Preparing an input file*

The `smart_parallel.sh` is the only script that the user will call. This script only needs one simple bash file as an entry, that should define the following variables:

- `initfile`: name of NAMD basic configuration file (`namd.conf` in this tutorial)
- `numinst`: number of AMS instances, i.e. the number of requested AMS runs.
- `outdir`: root directory to save all the AMS instances directories. If this directory exists, the script will look for results from previous run and will perform the missing AMS runs to complete `numinst` simulations.
- `parallel`: number of AMS instances that can be run in parallel.
- `numrep`: number of replicas for each AMS run (parameter N)
- `amstype`: `single` (all the replicas are initiated from the same point), `mult` (replicas are initiated from a set of `numrep` points) or `var` (the initial conditions will vary)
- `zmin`: minimum value for the reaction coordinate (only used if `amstype == var`)
- `zmax`: maximum value of ξ ($z_{\max}$)
- `timelimit`: simulation time limit in hours. This time limit prevents AMS runs from being abruptly killed if its duration exceeds a time limit from a queue system, and facilitates subsequent restart of the simulations afterwards.

- `icprefix`: prefix for `coor`, `vel` and `xsc` files from initial condition. If `amstype == mult` the files have to be named `prefix.`n (were $n = 0, ..., $`numrep` $- 1$).
- `zone`: name of Tcl script that contains the procedure `zone` (see `inzone.tcl` from the previous list).
- `measure`: name of Tcl script with the definition of the procedure `ams_measure` (see `coord.tcl`).
- `variables`: name of Tcl script with procedure `variables` (see `variables.tcl`).
- `amssteptime`: number of time steps between two computations of the reaction coordinate (this is the parameter K_{AMS} mentioned above).
- `tokill`: minimal number of replicas to kill at each iteration (parameter k)
- `getpaths`: on or off. If this variable is set to `on`, all the sampled trajectories will be given in text format files, built using the `variables` proc.
- `charmrunp`: number of processors to employ for the MD (if 0 the command will be `namd2`)
- `removefiles`: `yes` or `no`. If this variable is set to `yes`, all the AMS files will be removed after the run. If `getpaths == on`, all the trajectories will be obtained and will not be erased. Attention, if `getpaths == off`, and `removefiles == yes` it will be impossible to obtain the trajectories after the run.

2. Applying AMS to the alanine dipeptide isomerization in vacuum

We chose the alanine dipeptide isomerization in vacuum ($\mathrm{C_{eq}} \rightarrow \mathrm{C_{ax}}$ transition) to illustrate how to utilize the AMS method. The reader will find the precise definitions of regions A and B and of the reaction coordinate ξ in Section **2.1.**. The hands-on part of the tutorial starts in Section **2.2.**, where we show how to obtain the transition probability, starting all the replicas from the same point. In Section **2.3.** the theoretical underpinnings of the equation used to calculate the transition time using AMS results is given, as well as the guidelines as how to set up these simulations. Finally, armed with the results obtained in Sections **2.2.** and **2.3.** we will calculate the flux of reactive trajectories in Section **2.4.**.

2.1. Definitions of A, B and ξ

All the definitions will be based on the two dihedral angles φ and ψ (see Figure 2). The regions A and B are defined as two ellipses that cover the most significant wells on the free energy

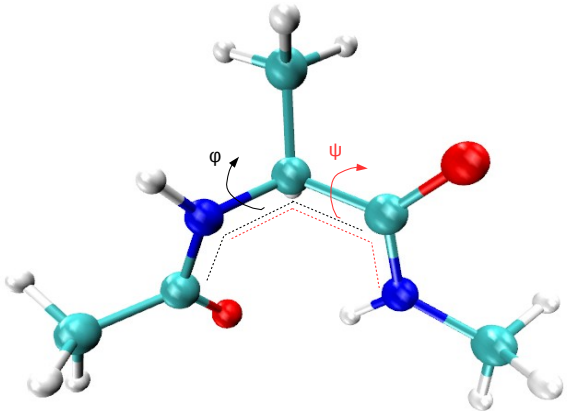


Figure 2: The dihedral angles φ and ψ used to distinguish between the C_{eq} and C_{ax} conformations.

landscape. The reaction coordinate is a measure of the distances from the two ellipses:

$$\xi(\varphi, \psi) = \min(d_A, 6.4) - \min(d_B, 3.8). \tag{3}$$

In equation (3), d_A (resp. d_B) is the sum of the Euclidean distances to the foci of the ellipse A (resp. B). The contour plot of the function ξ is given on Figure 4. We will employ $z_{\max} = 4.9$ in our simulations.

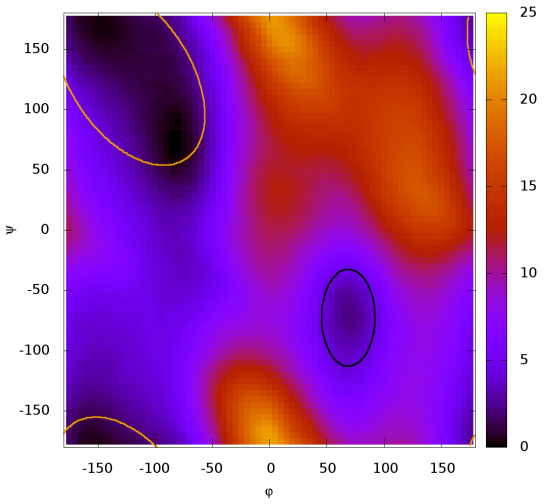


Figure 3: The free energy landscape [6] with the definition of zones A (yellow) and B (black).

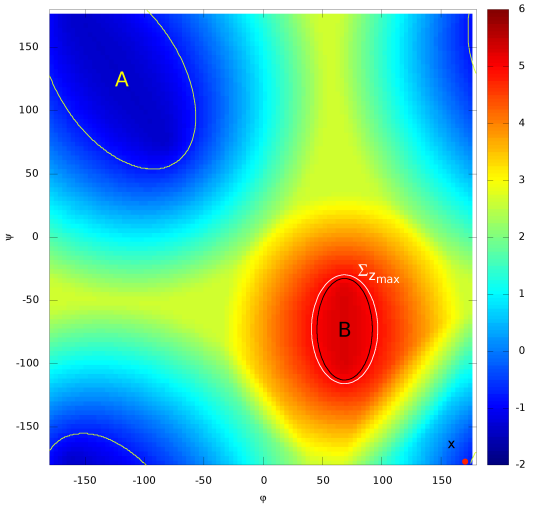


Figure 4: Contour plot of ξ, with regions A and B and the surface $\Sigma_{z_{\max}}$ ($z_{\max} = 4.9$).

2.2. Calculating the probability with AMS

In this section, we will see how to compute the probability to enter B before A, starting from a single fixed point. All the necessary files are located inside the directory `1-point`. For these simulations, the first set of replicas are trajectories that starts in a fixed point and finishes at A or B. This is indicated to the script `smart_parallel.sh` through the variable `amstype`, that should be set to `single`. We will start the replicas from the extended system and binary coordinate and velocity files with the prefix `point`.

1. Open the unfinished input file `1-point/point.par` to edit.
2. Set the auxiliary variable `path` with the path to all the tutorial files.
3. Set `amstype` to `single`. This means a simulation where all the replicas start from one single point.
4. Use the variable `icprefix` to give the prefix of the files from the starting point (`point` in this tutorial).

The results of this section will be used to calculate the flux in section **2.4.**. To obtain a net flux it is necessary to have approximately 9000 trajectories, and thus we need that `numinst` $\times$ `numrep` $>$ 9000. This is because in the case of extinction, no reactive trajectory will be sampled, so we overestimate the number of trajectories. In this tutorial, `numinst` $=$ 100 and `numrep` $=$ 100. We also have to tell our script to get the final trajectories, invited to so we can calculate the flux via the `getpaths` variable. If the reader is not interested in completing this tutorial and only want the calculation of the probability, set this variable to `off`.

5. Set `numinst = 100`.
6. Set `numrep = 100`.
7. Set the variable `getpaths` to `on`.

There is no special interest in obtaining the `dcd` trajectories for the alanine dipeptide case. Thus, to decrease disk space usage, all the files will be deleted at the end of the run.

8. Set the variable `removefiles` to `yes`.

The alanine dipeptide in vacuum is a really small system, so it is not necessary to run MD in parallel. However, we recommend running the AMS simulations in parallel and this should be adapted to the computer architecture at hand. Please, keep in mind that each one of the AMS simulations of this section takes about five minutes to complete (using 100 replicas). So a good estimation of the total time in minutes needed to complete all the 100 runs is:

$$\text{total time} = 5 \times \frac{\texttt{numinst}}{\texttt{parallel}}.$$

For example, using notebook with Intel core i7 processor, one can utilize `parallel = 8`, so the total time will be about one hour.

9. Set `charmrunp` to 0.
10. Set the `parallel` variable to the number of cores at hand.

Now, the final input file should look like this:

```
path="/path/to/tutorial/files"
outdir=$path"/1-point/ams"
tokill="1"
amstype="single"
numinst="100"
numrep="100"
zmax="4.90"
timelimit="240"
icprefix=$path"/1-point/point"
zone=$path"/common/inzone.tcl"
measure=$path"/common/coord.tcl"
variables=$path"/common/variables.tcl"
initfile=$path"/common/namd.conf"
amssteptime="20"
parallel="8"
getpaths="on"
charmrunp="0"
removefiles="yes"
```

Notice here that we are using `amssteptime = 20`. If the reader is guiding himself through this tutorial to run simulations with another system, be careful when choosing this parameter. First, using `amssteptime = 1` is always an option, but this will make the simulations slow. Second, if `amssteptime > 1`, it is necessary to satisfy one important condition: it should be small enough, so that if the system passes through A, at least one point inside A will be computed. Thus, we recommend to run a small preliminary simulation to evaluate the mean time the system stays inside A.

11. Run the script:
    `../smart/smart_parallel.sh point.par`

Running the script will block the screen showing what instance has already been launched. At the end of the run the probability estimation is given, as well as the total wall clock time spent, and other four files with the same name as the `outdir` variable, followed by:

- `cputime`: list of the CPU time of each AMS run
- `runtime`: same but with the wall-clock times
- `proba`: list of estimated probabilities
- `T3`: list of MD steps of the sampled reaction trajectories. This will be used in section **2.3.**.

The `smart` directory contains an executable file named `media`. The argument for this program is a file with numbers in one column, and their average value and standard deviation will be computed.

12. To see the final estimated value for the probability, type:
    `../smart/media ams.proba`

Compare the obtained result to the reference DNS value: $(2.076 \pm 0.357) \times 10^{-4}$.

Performing the simulation in this section using smaller values for `numrep` and/or `numinst` leads to a larger confidence interval. If the reader wish to make it smaller, it is possible to run the script again with a larger value of `numinst`. The script will not overwrite the previous results; instead it will run the remaining instances to complete the `numinst` AMS runs.

2.3. Obtaining the transition time using AMS results

As already mentioned, it is possible to calculate the transition time using the probability obtained with AMS by using a specific set of initial conditions, which we will now see how to obtain.

The transition time is the average time of the trajectories, coming from B, from its first entrance in A, until the first subsequent entrance in B [7, 8]. As A is metastable, the dynamics tends to make loops between A and its neighborhood before visiting B. To correctly define those loops,

let us fix an intermediate value $z_{\min}$ of the reaction coordinate, defining a surface $\Sigma_{z_{\min}}$ that corresponds to the region in which ξ is equal to $z_{\min}$.

If A is metastable and $\Sigma_{z_{\min}}$ is close to A, the number of loops made between A and $\Sigma_{z_{\min}}$ before visiting B will then large. After going through some of them, the system reaches an equilibrium. When this equilibrium is reached, the first hits of $\Sigma_{z_{\min}}$ follow a so-called quasi-stationary distribution μ_{QSD}. Here, we call the first hitting points of $\Sigma_{z_{\min}}$ the first points that, coming from A, have a ξ-value larger than $z_{\min}$. Using as an initial condition points distributed according to μ_{QSD}, it is possible to evaluate the probability p to reach B before A, starting from $\Sigma_{z_{\min}}$ at equilibrium with AMS. As A is metastable, the number of loops needed to reach the equilibrium will be small compared to the total number of loops followed before going to B. Thus, the time spent to reach the equilibrium can be neglected.

Let us now consider an equilibrium trajectory coming from B that enters A and returns to B. The goal is to calculate the average time ($\mathbb{E}(T_{AB})$) of this trajectory. A good strategy is to split this path in two: the loops between A and $\Sigma_{z_{\min}}$, and the reaction trajectory, i.e. the path from A to B that does not come back to A after reaching $\Sigma_{z_{\min}}$[8]. Neglecting the first time taken to go out of A, one can define as T^k_{loop} the time of the k^{th} loop between two subsequent hits of $\Sigma_{z_{\min}}$, conditioned to have visited A between them, and as T_{reac} the time of the reaction trajectory. If the number of loops made before visiting B is n, the time T_{AB} can be obtained as:

$$T_{AB} = \sum_{k=1}^{n} T^k_{\mathrm{loop}} + T_{\mathrm{reac}}. \tag{4}$$

At each passage over $\Sigma_{z_{\min}}$ there are two possible events: (i) first enter A, or (ii) first enter B. Using the probability p from the previous paragraph, the average number of loops before entering B is $1/p - 1$. This leads us to the final equation for the expected value of T_{AB}:

$$\mathbb{E}(T_{AB}) = \left(\frac{1}{p} - 1\right) \mathbb{E}(T_{\mathrm{loop}}) + \mathbb{E}(T_{\mathrm{reac}}). \tag{5}$$

Attention ! The calculations of this section needs several hours of computer time. This is due to the difficulty to correctly sample the initial conditions, as explained below. The reader following this tutorial in a NAMD hands-on workshop is invited to skip to Section **2.4.** and use the provided results for this section.

It has been shown that a good way to sample μ_{QSD} is to change the set of initial conditions at each run [9]. To do so, the user has to provide the value of $z_{\min}$. A small simulation before each AMS run is performed and the first `numrep` trajectories between $\Sigma_{z_{\min}}$ and A are used as the

first set of replicas. This is done just by setting the variable `amstype` to `var`. All the simulations will start from a point inside of A (files with prefix `A`).

The sampling of μ_{QSD} is not easy, and thus it is necessary to use more replicas and run more AMS simulations, compared with the simulations in Section **2.2.**), in order to get the desirable results. Go to the directory `2-time` for this part of the tutorial.

1. Copy the input file of the previous section and rename it `time.par`. A few editions are necessary.
2. Set the variable `amstype` to `var`.
3. Set the variable `numrep` to 500.
4. Set `numinst = 1000`.

First, it is necessary to provide the variable `zmin`. The choice of this parameter may be delicate. The closer $\Sigma_{z_{\min}}$ to A, the smaller the probability p to estimate. On the other hand, if $\Sigma_{z_{\min}}$ is too far from A, it will be harder to sample the loops between A and $\Sigma_{z_{\min}}$, and the underlying assumption of quasi-equilibrium before transiting to B will not be satisfied, which will imply a bias on the estimate of the transition time by formula (5). Moreover, the time needed in the initialization step will be larger. In this tutorial we will set `zmin = -0.6`, but we invite the reader to change this parameter and compare the results.

5. Set `zmin` to -0.6.
6. Change the variable `outdir`, otherwise the script will not run any new simulation.
7. Run the script:
   `../smart/smart_parallel.sh time.par`

When using `amstype` as `var`, the script will create two more output files: `ams.T1` and `ams.T2`. To obtain the transition time the user will run the provided program `ams_time` in directory `smart`. The argument for this program is a file that contains, in this exact order: the probability and the obtained values for T_1, T_2 (whose sum is equal to $\mathbb{E}(T_{\mathrm{loop}})$) and T_3 (equal to $\mathbb{E}(T_{\mathrm{reac}})$). All of these values have to be provided with the confidence interval and it is possible to obtain them utilizing the executable file `media`, just like in the previous section.

8. Run this command line with files `ams.proba`, `ams.T1`, `ams.T2` and `ams.T3` (in this exact order), and redirect the output in a file named `for_time`.
   `../smart/media ams.proba >> for_time`

9. Run the following command line:
   `../smart/time_ams for_time`

Compare the obtained result to the reference value of: (309.5 ± 23.8) ns.

2.4. Calculating the flux of reactive trajectories sampled with AMS

Using a set of reaction trajectories obtained with the AMS method, each trajectory i can be associated with a vector $(\boldsymbol{\theta}_t^i)_{t\in[0,\tau_B^i]}$ with the two dihedral angles at each point. The (φ, ψ) space is split into L cells $(C_l)_{1\leq l\leq L}$. The flux of reactive trajectories in each cell is then defined up to a multiplicative constant by (compare with equation of Remark 1.13 in reference [7]):

$$J(C_l) = \sum_{i=1}^{n} \sum_{t=0}^{\tau_B^i - 1} \left(\frac{\boldsymbol{\theta}_{t+1}^i - \boldsymbol{\theta}_t^i}{\Delta t} \right) \mathbb{1}_{\boldsymbol{\theta}_t^i \in C_l}. \tag{6}$$

The parameter L should be given by the user. In this tutorial $L = 50 \times 50$.

A program that calculates the reactive trajectories flux using the expression above is provided in the `smart` directory. The user only needs to provide a file containing the list of files with the trajectories sampled by AMS. Such a file is actually given by the `smart_parallel.sh` script, and the user should find it inside the `outdir` directory under the name `paths_list`. Please note that the provided program only calculates the flux in two dimensions.

1. In the terminal, `cd` directory `3-flux`
2. Calculate the flux with the results from Section **2.2.**
   `../smart/flux ../1-point/ams/paths_list point.flux 50 20`
   The last value corresponds to the `amssteptime`.
3. The same for Section **2.3.**
   `../smart/flux ../2-time/ams/paths_list time.flux 50 20`

If the reader is performing only this Section of the tutorial, use the provided results located in directory `example_results`. In this case the reader will need to modify the full paths listed in `3-flux/example_results/2-time/path_list`.

`The flux file contains five columns, that corresponds to the vector position, the vector direction (unit vector) and size. The files point.flux and time.flux can then be plotted using the program the user prefers. If the`

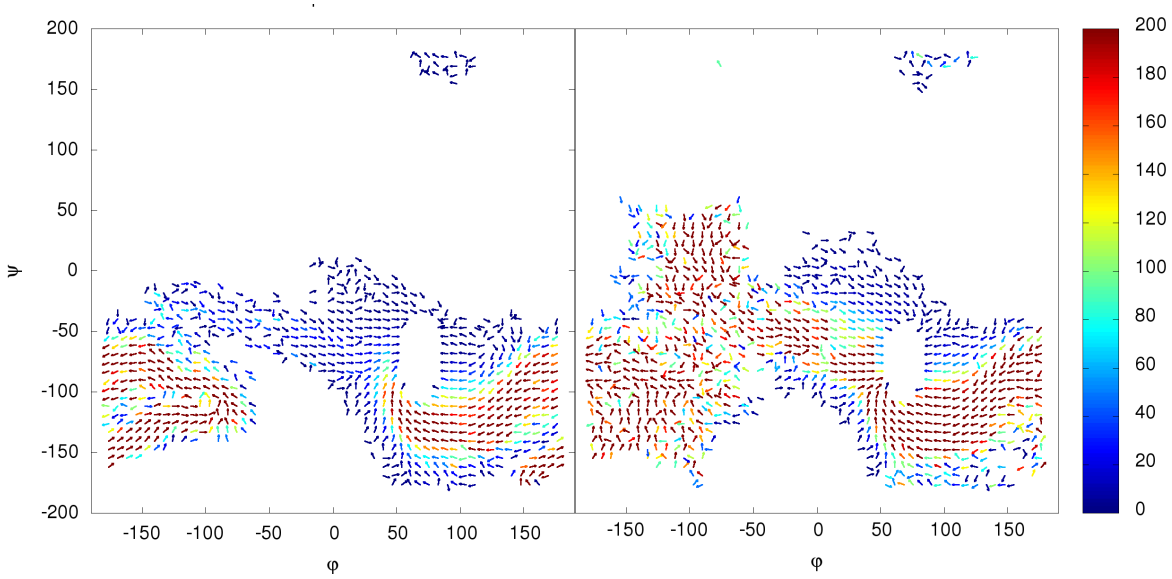


Figure 5: Example of results for the flux of reactive paths.

reader has access to Gnuplot program, we provided a script file, named make_plot, used to make Figure 5.

4. In side directory 3-flux, type:
   gnuplot make_plot.

5. Change the variable cutoff and repeat the previous step until a desirable result is achieved.

The flux of reactive trajectories can give an idea of the preferable paths from A to B. They strongly depend on the initial conditions. Notice that time.flux was calculated using variable initial conditions created by sampling loops between A and $\Sigma_{z_{\min}}$, as explained in Section **2.3.**. Thus, if corresponds to the flux of reactive trajectories at equilibrium.

References

[1] F. Cérou and A. Guyader. Adaptive multilevel splitting for rare event analysis. *Stoch. Anal. Appl.*, 25(2):417-443, 2007.

[2] Frédéric Cérou, Arnaud Guyader, Tony Lelièvre, and David Pommier. A multiple replica approach to simulate reactive trajectories. *J. Chem. Phys.*, 134(5):054108, 2011.

[3] Aristoff, David and Lelièvre, Tony and Mayne, Christopher G and Teo, Ivan Adaptive multilevel splitting in molecular dynamics simulations *ESAIM: Proceedings and Surveys*, 48:215-225, 2015.

[4] I. Teo, C. G. Mayne, K. Schulten and T. Lelièvre. Adaptive Multilevel Splitting Method for Molecular Dynamics Calculation of Benzamidine-Trypsin Dissociation Time. *J. Chem. Theory Comput.*, 12(6):2983-2989, 2016.

[5] G. Fiorin, M. L. Klein, and J. Hénin. Using collective variables to drive molecular dynamics simulations. *Mol. Phys.*, 111(22 - 23):3345 - 3362, 2013.

[6] J. Hénin, Giacomo F., C. Chipot, and M. L. Klein. Exploring multidimensional free energy landscapes using time-dependent biases on collective variables. *J. Chem. Theory Comput.*, 6(1):35-47, 2010.

[7] J. Lu and J. Nolen. Reactive trajectories and the transition path process. *Probability Theory and Related Fields*, 161(1-2):195-244, 2015.

[8] E. Vanden-Eijnden. Transition path theory. *Lect. Notes Phys.*, 703:439-478, 2006.

[9] L. J. S. Lopes and T. Lelièvre. Analysis of the adaptive multilevel splitting method with the alanine di-peptide's isomerization. *arXiv:1707.00950 [physics.chem-ph]*, 2017.