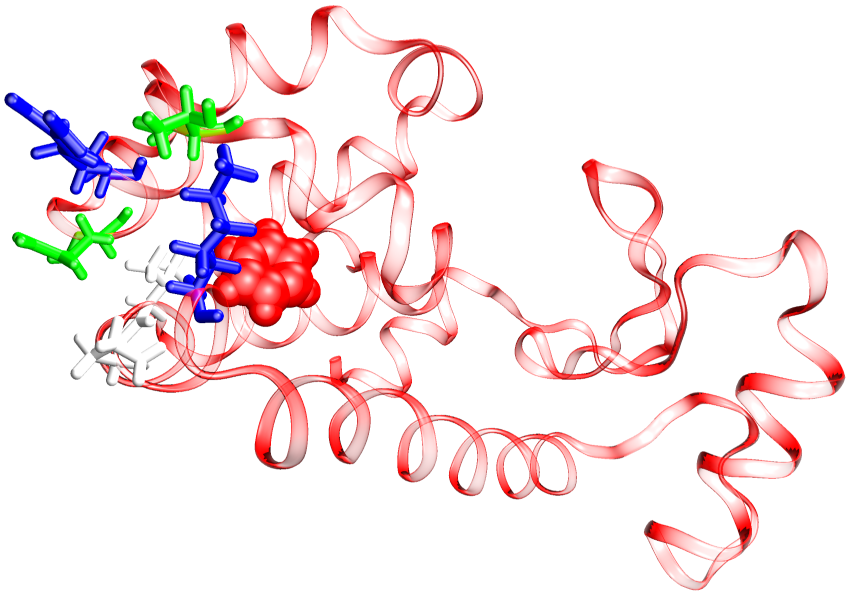
# Steered Molecular Dynamics with Tinker–HP TUTORIAL



⚠ Please visit the website: https://github.com/TinkerTools to obtain last updates of this tutorial and informations about other new tutorials.

# Contents

# 1 Introduction

The aim of this tutorial is to introduce the use of the Steered Molecular Dynamics within the Tinker–HP package.
By applying an external force to the system we can induce rare events occurring in a frame of few nanoseconds instead of several microseconds. It is therefore useful to enhance the conformational sampling of a system such as:

- Scanning of an unfolding pathway
- Docking/Undocking of a ligand out of an enzyme's binding pocket
- Mimicking AFM study
- Preparing Umbrella Sampling Windows
- ...

In the context of this tutorial we decided to illustrate the use of SMD in Tinker–HP by inducing the exit process of a benzene molecule out of the buried binding pocket of a lyzosyme as represented on Figure 1. The system, coming from PDB 181l, has been already prepared as it was converted into a txyz file, solvated and neutralized according to the procedure described in our cMD tutorial in Tinker–HP. Files are ready to be used in a SMD simulation. Therefore, we deemed at providing along this tutorial an easy manner of how to induce the exit of the benzene molecule out of our system using the CVSMD procedure. Tools for a CFSMD is also depicted and explained.

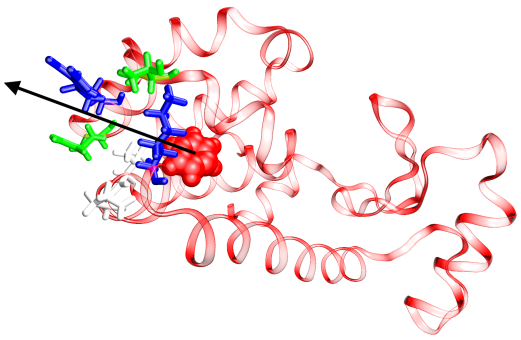


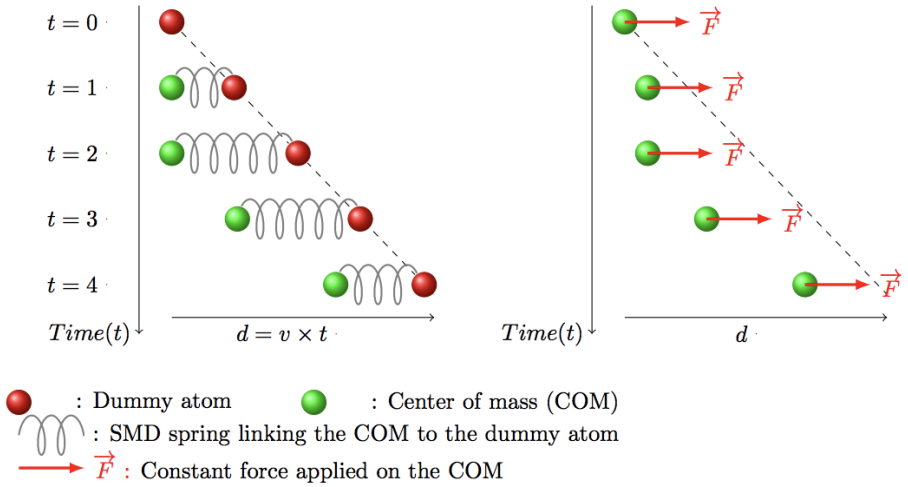
Figure 1: Representation of the lyzosyme system (PDB: 181l) containing a benzene molecule (in red) in its active site. Residues at the exit have been drawn in green, blue and white while the black arrow indicates the steering direction as it corresponds to the vector formed by the center of mass (COM) of the benzene molecule and the depicted residues at the exit of the binding pocket.

# 2 SMD simulation

## 2.1 SMD theory

Several options exist to observe rare events in MD, such as:

- Run a simple and equilibrium MD, but it probably takes a too much long time of simulation to observe the desired event
- Run a non–equilibrium MD, which induce the breakage of the beta–strands by steering it in a specific direction of the space. The time of simulation will decrease a lot.




To steer one or more than one atom (COM) in a specific direction of the space, the Steered Molecular Dynamic (SMD) methodology appears to be the best to use. As depcited on the upper Figure, two kinds of SMD could be performed:

1. Constant–Velocity SMD (CVSMD): a harmonic spring is attached between one or several atoms (namely COM) and a dummy atom. While the dummy atom run at constant velocity, it steers with it the COM, finally inducing the desired motion. During the simulation, the COM is thus restrained by a harmonically potential:

$$U(\vec{r_1}, \vec{r_2}, ..., \vec{r_n}) = \frac{1}{2}k(vt - (\vec{R(t)} - \vec{R_0}).\vec{n})^2 \quad (1)$$

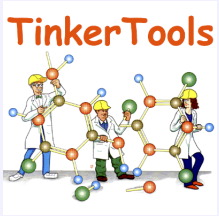
   Where k is the force constant of the spring, v the SMD velocity, $\vec{R_0}$ the initial position of the COM and $\vec{n}$ a normalized vector corresponding the motion's direction. Note that a second spring $k_2$ could be added to the system to avoid fluctuations. The harmonically potential becomes:

$$U(\vec{r_1}, \vec{r_2}, ..., \vec{r_n}) = \frac{1}{2}k(vt - (\vec{R(t)} - \vec{R_0}).\vec{n})^2 + \frac{1}{2}k_2((\vec{R(t)} - \vec{R_0})^2 - ((\vec{R(t)} - \vec{R_0}).\vec{n})^2) \quad (2)$$

2. Constant–Force SMD (CFSMD): No harmonic spring or dummy atom are used here. We just applied a constant force on one or several atoms during the MD.

## 2.2 Constant–Velocity SMD (CVSMD)

**Input Files and Executables**




- system.xyz
- system.key
- AMOEBABIO18.prm
- Tinker–HP dynamic

To prepare the system for a CVSMD simulation, create a folder and place in the files system.xyz, system.key and the executable dynamic. The system.key file used to provide the SMD instructions for Tinker–HP. The system.key file is similar as the one used for the cMD of ubiquitin in another tutorial. We just add here the CVSMD instructions. Let us take a look on the different options mandatory and optional for using CVSMD.

- **CVSMD**
  Importance = mandatory
  Default value: –
  Possible values: –
  Meaning: keyword which activate the CVSMD module in Tinker–HP.

- **SMDk**
  Importance = mandatory
  Default value: 0.00 kcal/mol.Å$^2$
  Possible values: 1 positive real
  Meaning: specify the force constant of the spring.
  Unity: kcal/mol.Å$^2$

- **SMDk2**
  Importance = optional
  Default value: 0.00 kcal/mol.Å$^2$
  Possible values: 1 positive real
  Meaning: specify the force constant of the transverse spring.
  Unity: kcal/mol.Å$^2$

- **SMDvel**
  Importance = mandatory Default value: 0.00 Å/ps
  Possible values: 1 positive real
  Meaning: specify the SMD velocity apply on the COM.
  Unity: Å/ps

- **SMDatoms**
  Importance = mandatory
  Default value: 0
  Possible values: 1 positive integer + other positive integers (or -1 integer and 1 integer)
  Meaning: the first integer specify how many atoms are assigned to the COM. Each other integers indicates the index of these atoms according to the using txyz file. The

quantity of the "other positive integers" has to be equal to the first positive integer. If for instance 100 atoms want to be specified, ranging from 123 to 222, it can be indicated as 100 -123 222.

- **SMDdir**
  Importance = mandatory
  Default value: 0.000 0.000 0.000
  Possible values: 3 reals
  Meaning: specify the direction vector of the SMD. Note that it is not necessary to normalize it here, while Tinker–HP will make it automatically.

- **ManualCOM**
  Importance = optional
  Default value: –
  Possible value: 3 reals
  Meaning: Useful when we want to manually place the dummy atom at the starting point of our dynamics.

- **SMDoutputfreq**
  Importance = optional
  Default value: 1
  Possible value: 1 positive integer
  Meaning: specify at which frequency of timestep we want to print out the SMD results in the SMD output file.

- **SMDdt**
  Importance = optional
  Default value: 0.001 Å/ps
  Possible values: 1 positive real
  Meaning: specify the timestep to take into account in the SMD procedure.
  Unity: ps

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
###################################
############## EXTRA ##############
###################################
archive
printout                1000                    # Interval at which to print out energies
inactive                2 845 1964 2582         # Atoms 2 845 1964 and 2582 are frozen
tortorterm              none
#
########## SMD block #############
CVSMD                                       # Use of the constant velocity SMD method
SMDk                    7.00                # k constant of the spring1 (Kcal/mol/A**2)
SMDvel                  0.01                # SMD velocity (A/ps)
SMDatoms                12 -2604 2615       # number of atoms assigned to the COM (12) and their indexes
SMDdir                 -4.834 5.513 -2.323  # direction vector of the COM motion (not normalized)
SMDoutputfreq           100                 # frequency of the SMD results (in fs)
SMDdt                   0.010               # time step incrementation in the forces calculations (in ps)
#########################################
```

Figure 2: Representation of the SMD block in the system.key file.

We depicted an example in Figure 2 about what is must look like in the key file. In this case, the CVSMD keyword is on the ON mode, with a spring constant k equal to 7.00 kcal/mol.Å$^2$.

A SMD velocity of 0.01 Å/ps has been chosen, applied on the center of mass (COM) of 12 atoms, ranging from 2604 to 2615. The steering direction is defined by the vector (-4.834 5.513 -2.323). The timestep of the dynamics is set to 10 fs (0.010 ps) and each SMD data will be printed every 100 steps (so at each 1 fs in our case due to the 10 fs timestep used here).
Launch a SMD simulation is similar as a cMD. While our SMD velocity is fixed at 0.01 Å/ps and we want to perform a steering process on 20 Å, a simulation time of 2000 ps is needed. Type in the terminal the following command:

```
mpi -np 8 ./dynamic system 200000 10 1 4 300 1 > system.out
```

In addition to create a arc, out and dyn files a SMD output file will be also generated and filled according to the parameters specified before in the key file.

**3D spatial decomposition limitation in SMD ...**

As we are using the 3D spatial decomposition with midpoint Method in Tinker–HP, each core knows the x/y/z coordinates of their respective atoms + atoms localized at half the maximum cutoff size. In order to ensure the consistency of our SMD procedure, we need to be sure that all the SMD atoms could be read by at least one core, especially when more than one atom (COM) is used. In this way, one possibility is to consider the cutoff range equal to the radius of gyration formed by all the SMD atoms. You just then have to enter a keyword labeled **dd-cutoff**.
Example: We have a molecule of 25 atoms with a radius of gyration equal to 10 Å. You just enter in your key file **dd-cutoff 10**. It will decrease the speed of the calculation but ensures the fiability of the SMD procedure.

**Output Files**

- system.out
- system.arc
- system.dyn
- SMD_output.dat

## 2.3 Constant–Force SMD (CFSMD)

**Input Files and Executables**


- system.xyz
- system.key
- AMOEBABIO18.prm
- Tinker–HP dynamic

Preparing a system for a CFSMD simulation is similar to what we show before for the CVSMD case. As before let us take a look on the different mandatory and optional options to use in the key file in the CFSMD case.

- **CFSMD**
  Importance = mandatory
  Default value: –
  Possible values: –
  Meaning: keyword which activate the CFSMD module in Tinker–HP.

- **SMDfor**
  Importance = mandatory Default value: 0.00 Å/ps
  Possible values: 1 positive real
  Meaning: specify the SMD constant force applied on the COM.
  Unity: kcal/mol.Å$^2$

- **SMDatoms**
  Importance = mandatory
  Default value: 0
  Possible values: 1 positive integer + other positive integers
  Meaning: the first integer specify how many atoms are assigned to the COM. Each other integers indicates the index of these atoms according to the using txyz file. The quantity of the "other positive integers" has to be equal to the first positive integer. Same possibility as explained for CVSMD is possible if a lot of atoms have to be declared.

- **SMDdir**
  Importance = mandatory
  Default value: 0.000 0.000 0.000
  Possible values: 3 reals
  Meaning: specify the direction vector of the SMD. Note that it is not necessary to normalize it here, while Tinker–HP will make it automatically.

- **SMDoutputfreq**
  Importance = optional
  Default value: 1 Possible value: 1 positive integer
  Meaning: specify at which frequency of timestep we want to print out the SMD results in the SMD output file.

- **SMDdt**
  Importance = optional
  Default value: 0.001 Å/ps
  Possible values: 1 positive real
  Meaning: specify the timestep to take into account in the SMD procedure.
  Unity: ps

We depicted an example in Figure 3 about what is must look like in the key file. In this case, the CFSMD keyword is on the ON mode, with a force constant equal to 5.00 kcal/mol.Å$^2$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
###################################
############### EXTRA ###############
###################################
archive
printout                1000                        # Interval at which to print out energies
inactive                2 845 1964 2582             # Atoms 2 845 1964 and 2582 are frozen
tortorterm              none
#
########## SMD block #############
CFSMD                                           # Use of the constant velocity SMD method
SMDFor                  5.00                    # Constant force (in kcal/mol.A^2)
SMDatoms                12 -2604 2615           # number of atoms assigned to the COM (12) and their indexes
SMDdir                 -4.834 5.513 -2.323      # direction vector of the COM motion (not normalized)
SMDoutputfreq           100                     # frequency of the SMD results (in fs)
SMDdt                   0.010                   # time step incrementation in the forces calculations (in ps)
#########################################
```

Figure 3: Representation of the SMD block in the system.key file.

applied on the center of mass (COM) of 12 atoms, ranging from 2604 to 2615. The steering direction is defined by the vector (-4.834 5.513 -2.323). The timestep of the dynamics is set to 10 fs (0.010 ps) and each SMD data will be printed every 100 steps (so at each 1 fs in our case due to the 10 fs timestep used here).
Launch a CFSMD simulation is similar as before. As for the CVSMD, we will perform a simulation of 2000 ps. Please type in the terminal command:

*mpi -np 8 ./dynamic system 150000 1.0 1 2 300 > system.out*

In addition to create a arc, out and dyn files a SMD output file will be also generated and filled according to the parameters specified before in the key file.

**Output Files**

- system.out
- system.arc
- system.dyn
- SMD_output.dat

# 3 Analysis of a SMD trajectory

The final point of this tutorial is to show how we can analyze the SMD results provided by a Tinker-HP trajectory.
According to the literature, three types of graphic should be interesting to be construct:

1. The **pulling distance in function of the simulation time** (CVSMD and CFSMD)
2. The **steering force acting on the SMD atom(s) still in function of the simulation time** (CVSMD only)
3. The **pulling work in function of the pulling distance** (CVSMD only)

## 3.1 The SMD_output.dat format file

As explained before, most of the SMD data are collected during the dynamics in a specific file called SMD_output.dat. This file ensures to be able to follow the behavior of the SMD process independently of the main output file of Tinker–HP. Indeed, compared to other softwares such as NAMD for instance, it facilitates the post–treatment of the data once the dynamics is over.
Let explain now how the specific SMD output file is constructed:

1. After a designed promo a warning session is depicted. This one prevent to such eventual oversights (for instance it misses an SMDvel declaration in the key file so the SMDvel is set to be 0). If a mistake is present in the keyfile, an error message appears with a dead-skull meaning that your SMD calculation is over.
2. Initial parameters read in the keyfile are then listed in order to see if all of them are ok. A specific list of all the SMD atoms are also depicted with the initial SMD vector read in the keyfile and applied on the COM of these atoms.
3. To finish if no manual COM has been declared with the *ManualCOM* keyword, the COM is automatically calculated according to the declared SMD atoms previously. The SMD direction is also normalized if it necessary, so it is not mandatory to manually normalized it in the keyfile.
4. Once all the parameters are good, all the SMD data are collected in specific columns with a "SMD" word at the beginning of each lines. Although the first column only corresponds to the "SMD" word, we have:
   - $2^{nd}$ column: Time of the dynamics (in ps)
   - $3^{rd}/4^{th}/5^{th}$ columns: x/y/z coordinates of the COM
   - $6^{th}/7^{th}/8^{th}$ columns: x/y/z force components according to the CVSMD procedure. Note that these 3 columns are not present in the CFSMD one as the force is set to be constant.
5. Finally, to be sure that the SMD calculation is ended normally, an ended message depicts the well ended of the SMD procedure.

Now the SMD_output.dat structure is explained, let see how we can treat it to obtain the interested curves to analyze the SMD results.

## 3.2 First type of curve: The COM distance in function of the simulation time (CVSMD anf CFSMD)

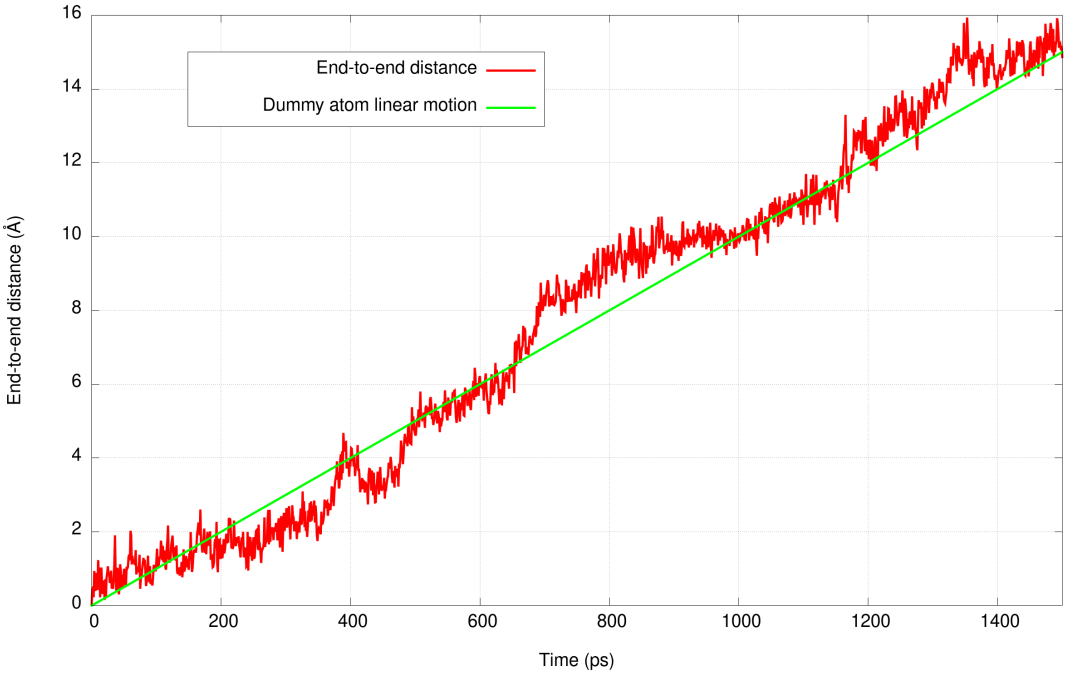
**Input Files**


- SMD_output.dat
- gnuplot
- distance_vs_time.gnu

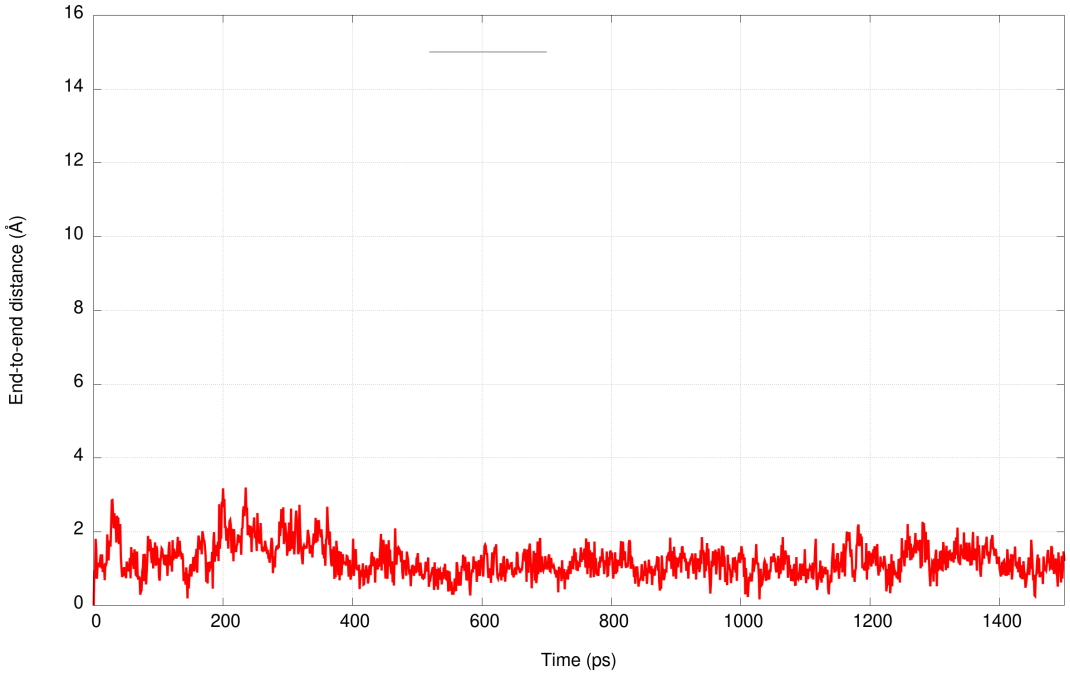
These graphs are not complicated to be plotted as all the informations are not to be modified in the SMD_output.dat file. Place the SMD_output.dat and distance_vs_time.gnu files in the same direction. Type in the terminal:

*gnuplot distance_vs_time.gnu*

For the CVSMD, you should obtain a new picture named Time_distance.png and look like:




For the CFSMD, make the same procedure and the Time_distance.png obtained is:




**Interpretation tool !**

- The dummy and COM of the SMD atoms follow the same path from the beginning of the simulation.
- A constant force of 5 kcal/mol.Å$^2$ is not sufficient to induce the undocking process of the benzene molecule. Either the constant force should be increase or the simulation time.

## 3.3 Second type of curve: The steering force acting on the SMD atom(s) in function of the simulation time (CVSMD only)

**Input Files**

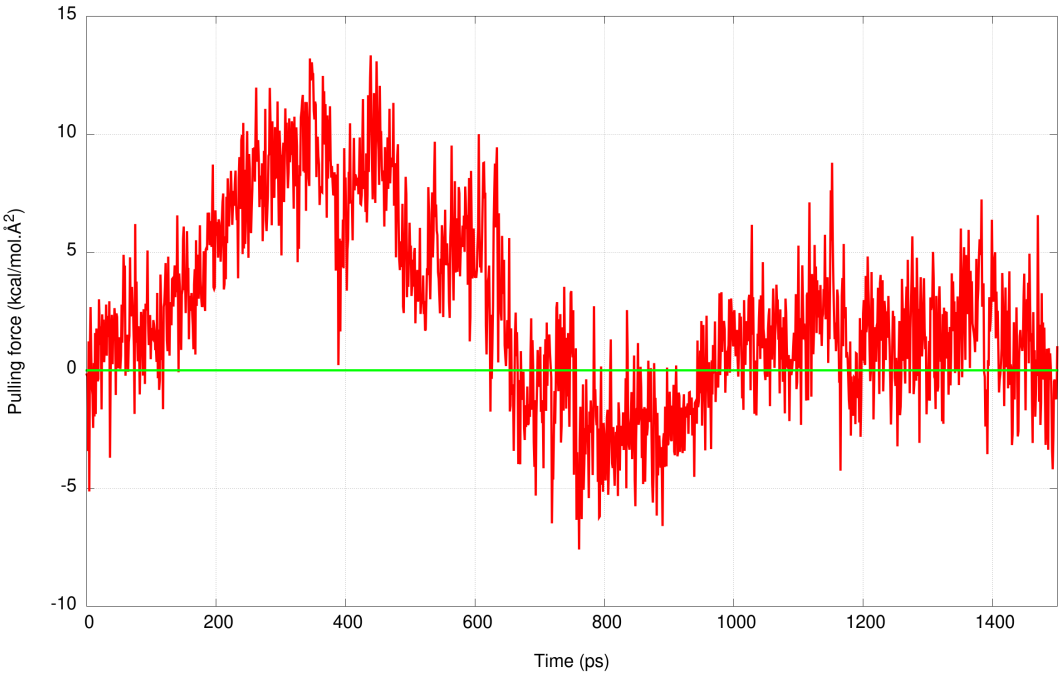

- SMD_output.dat
- gnuplot
- force_vs_time.gnu

As for the previously subsection, this graph is also not complicated to be plotted for the same reasons. Follow these instructions:

- Make in a same directory the SMD_output.dat and force_vs_time.gnu files
- Type in the terminal:

*gnuplot force_vs_time.gnu*

You should obtained the following graph if all is normally working in a file labeled Force_vs_time.png:




**Interpretation tool !**

- The green line corresponds to the 0 frontier between positive and negative forces
- The red curve corresponds to the pulling force applied on the COM of the SMD atoms.
- On the red curve, we reach a plateau of approximately 10 kcal/mol from 300 to 500 ps. This plateau can be directly linked to the CFSMD used additionally to the CVSMD as 5.00 kcal/mol.Å$^2$ could induce the undocking event of the benzene but with a slow manner. We suggest in this direction to consider the CFSMD as a useful tool to make preliminary tests on more complex systems or reaction coordinates. It could also be used as a tool to scan the metastability of systems where different constant forces could be use in several independant trajectories.
- The slope's fall of the red curve at 500 ps depcits the end of the undocking process, which is followed by a relaxation of the structure (and can also be observed directly by visualization of the SMD trajectory).

## 3.4 Third type of curve: The integration of the pulling forces along the time provides the pulling work in function of the distance (CVSMD only)

**Input Files**


- SMD_output1.dat
- gnuplot
- work.tcl
- work.gnu

Once the pulling force is calculated in function of the time, the last step is to be able to integrate it along the time. It thus provides what we call the pulling work profile of the SMD trajectory.
Starting from the SMD_output.dat file as the two previous subparts we are going to calculate the pulling work. To perform this step we suggest to use an already written script called work.tcl. A picture of it is given in Figure 4. The only part to be changed by the user is

```
#!/bin/tclsh

############################################################
### PULLING FORCES INTEGRATION TCL SCRIPT FOR TINKER-HP ##
############################################################
#
## Owner: FREDERIC CELERSE
## Institution: SORBONNE UNIVERSITE
## Last update: 29/10/2019
#

#### Open the log file for reading and the output .dat file for writing
set file [open SMD_output1.dat r]
set output [open work.dat w]
##### Parameters input from user.
set nx -0.628
set ny 0.717
set nz -0.302
set v 0.01
set dt 0.010
set outputfreq 100
set initdist 0
##### DO NOT MOVE FROM HERE !
set work 0.0
while { [gets $file line] != -1 } {
##### Determine if a line contains "SMD ". If so, write the
##### timestep followed by the calculated pulling work from
##### the output file
        if {[lindex $line 0] == "SMD"} {
                set work [expr $work + $v*$dt*$outputfreq*($nx*[lindex $line 5] + $ny*[lindex $line 6] + $nz*[lindex $line 7])]
                puts $output "$initdist $work"
                set initdist [expr $initdist + $v*$dt*$outputfreq]
        }
}
##### Close the log file and the output .dat file
close $file
close $output
```

Figure 4: Representation of the work.tcl script

localized in the "Parameters input for user". It encompasses:

- The **normalized** SMD vector nx/ny/nz (provided in the SMD_output.dat file)
- The SMD velocity v
- The timestep of the dynamics dt

- The frequency of printing in the SMD_output.dat file outputfreq
- The initial distance initdist

To apply this script, make sure that work.tcl and SMD_output.dat files are localized at the same directory. Once done please type in the terminal:

*tclsh work.tcl*

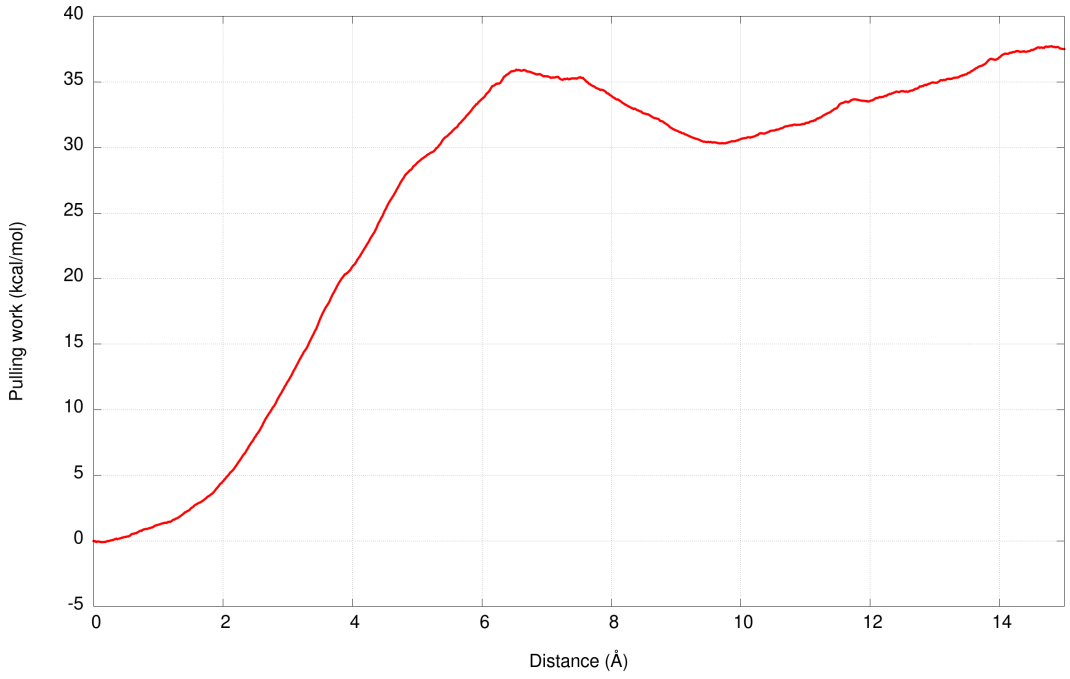
It creates a new file called SMD_sumoutput1.dat. This file is in two columns:

- First column: The current distance of pulling according to the dummy atom motion
- Second column: Its related pulling work

Finally, to plot the graph of the pulling work please use the work.gnu file and type in the terminal:

*gnuplot work.gnu*

You finally obtain this graph:




**Interpretation tool !**

In this trajectory, benzene needs to obtain a free energy of 36 kcal/mol to be undocked ! What about the validity of this value ?

# 4 Complementary informations

## 4.1 Beyond a simple SMD trajectory: the use of the Jarzynski equality

In this tutorial, only one trajectory has been performed. However, it is not appropriate to estimate properly the final free energy profile, especially on large scale systems with high SMD velocity. In this direction, we should be able to make a link between a set of non–reversible SMD trajectory and equilibrium properties such as the PMF. The Jarzynski equation is designed to tackle this problem, by linking the exponential of the free energy barrier $\Delta F$ with the average of the exponential of each of the N pulling works $W_i$ as follow:

$$exp(-\Delta F) = \langle exp(-\beta W_i) \rangle_{i=1,N} \tag{3}$$

The main principle is the following one: bigger N is, better is the exponential average. However, the main problem of this equation is localized in the number of trajectories to choose with a specific SMD velocity to reach a good convergence in term of free energy barrier. In most of the cases, the set N of trajectories is too big to be directly applicable on large scale systems from 10 000 to 1 000 000 atoms. In front of this situation, we invite the users to follow these specific paths of research:

- If you are in confidence that your rare event owns as we call a gaussian behavior, you can directly approximate your Jarzynski equality by applying the stiff spring approximation coupled to the second order of the cumulant expansion method.
- If you are not sure about the behavior of your rare event, we just invite you to simply decrease the SMD velocity as well as possible, and to couple the SMD module with new generations of algorithms present in Tinker-HP (such as the multitimestep BAOAB integrator or the TCG algorithm to accelerate the convergence of your dipole moments).
- Coupled to these acceleration processes, you can also pass from a unidirectional SMD to a bidirectional SMD, by just applying the reverse process at the end of you unidirectional SMD procedure. By using of the Crook's theorem or BAR, it should enhances the accuracy of your final free energy barrier, as well as it will also increase you simulation time when you have to double your total simulation time.

Script to use the Jarzynski equality and the cumulant expansion approach at the second order are provided with the tutorial as "Jarzynski_equality.f90" and "CE2.f90".

## 4.2 Other possible exercises

There is a list (not exhaustive) of possible exercises in the continuity of this tutorial.

**Exercise 1:** Estimate the simulation time to perform a set of 20 trajectories with a SMD velocity of: 1.0 Å/ps, 0.1 Å/ps, 0.01 Å/ps and 0.001 Å/ps.
**Exercise 2:** Make a set of 10 trajectories with a SMD velocity of 0.1 Å/ps and estimate the free energy barrier by using the Jarzynski equality.
**Exercise 3:** Same order as exercise 3 but with the stiff spring approximation and the second order of the cumulant expansion method. Compare the difference between the two curves.