Abin - *Ab initio* molecular dynamics program v1.0

Daniel Hollas

October 20, 2014

Contents

1 Introduction

ABIN is an program performing *ab initio* Born-Oppenheimer molecular dynamics. It was designed specifically to deal with quantum nuclear effects. It can do path integral simulations and also utilizes recently developed quantum thermostat[11]. It can also simulate non-adiabatic events using Surface-hoping algorithm.

The basic philosophy of ABIN is simple. While the program itself handles the propagation of the system according to the equations of motion, the forces and energies are taken from the external electronic structure program such as Gaussian. The call to the chosen external program is handled via simple shell script. Therefore, writing a new interface is rather straightforward and can be done without any changes to the code itself.

The current capabilities of ABIN are:

- Classical MD in the gas phase or in condensed phase using spherical boundary conditions. Both NVE and NVT ensembles are supported. The temperature is regulated using Nosé-Hoover chain thermostat[1]. Equations of motions are integrated using velocity Verlet algorithm. Forces and energies are obtained either within the code using classical Amber force field or from external programs. At this moment there are interfaces to the following programs: GAUSSIAN 09[2], MOLPRO[3], GAMESS[4], TURBOMOLE[5], QCHEM[6], ORCA[7] and TERACHEM[8].
- Path Integral MD using staging transformation, RESPA integrator and massive Nosé-Hoover Chain thermostat.
- Surface hopping algorithm[9] for mixed quantum classical MD. This code is currently interfaced with MOLPRO and GAMESS packages.
- Simulations with quantum thermostat[10] and combined PI+GLE[11] methods.
- two-layer QM/MM ONIOM interface, which calculates the "MM" part using any external program.
- Constraints using SHAKE algorithm (currently only for classical MD).
- Simple minimization using the steepest descent method.

Several other capabilities are currently under development, namely:

- PIMD utilizing normal mode coordinates, which will allow for PIMD simulations with constraints. Normal modes are also important for an efficient implementation of the PI+GLE method.
- Periodic boundary conditions, allowing for the simulations of condensed phases (at the molecular mechanical level).
- Ewald summation algorithm for calculation of long-range coulomb forces and energies.

1.1 Installation

ABIN is distributed in the form of Fortran and C source code and is intended and tested to be used in Linux environment. In the rest of the manual, it is always assumed that you are in the top ABIN directory (i.e. ABINv1.0). You should see the following content:

src/ Directory with source code.

ABIN-INTERFACES/ Directory with interfaces for many common *ab initio* codes.

bin/ Directory containing the compiled binary *abin.v1.0* and other small handy scripts. It is highly recommended to copy the files from bin/ directory somewhere to your $PATH (i.e. ∼/bin) Some of scripts need to be in your path in order for other utilities in the UTIL/ directory to work.

UTIL/ Directory containing many scripts and small programs that allow efficient usage of Abin . See UTIL/README for a brief overview.

SAMPLE/ Directory with sample input files.

To compile the code, you need the following to be installed in your system:

- gcc and gfortran compilers, version at least 4.3, although versions ¿=4.6 are preferred. In the improbable case that your system has older version, you can download gfortran binaries here:

  https://gcc.gnu.org/wiki/GFortranBinaries

- FFTW library, which is typically provided with your linux distribution or it can be downloaded here:

  http://www.fftw.org/download.html

The compilation can be as easy as:

$ *cd src*; *make*

You can test the installation by:

$ *make test* (one of the test requires a working Orca installation)

Before recompilation, you should always clean up by:

$ *make clean*

2 Quick start

2.1 Input files

There are two mandatory input files – one contains the input coordinates in xyz format and the other one contains all the parameters specifying the simulation.

The xyz format is specified as follows: first line contains number of atoms, second line is empty or contains some comment and the following lines contains cartesian coordinates (in Ångstroms) in the form "*atom-name x y z*". The atom names are not case-sensitive.

The parameter file contains all parameters needed for the simulation. The input uses Fortran namelist syntax. Variables are grouped into different namelists and there are two namelists that have to be always present: *general* and *nhcopt*. For each type of simulation, there is an example input file in the folder SAMPLES/. Here is the minimum input file for classical MD.

```
-----------------------------------------------------------------
Everything outside namelists is ignored

&general                ! This is the beginning of a namelist.
!Comments within a namelist start with exclamation mark.
```

```
!The order of keywords is irrelevant.
!Most of the keywords have reasonable default values.
pot='g09'               ! where do we get forces and energies?
natom=27,               ! number of atoms
ipimd=0,                ! 0=CMD, 1=PIMD,2=SH,3=minimization
mini=2000,              ! equilibration period (should be at least 2000)

!keywords on one line are separated by a comma
nstep=50000,dt=20.,     ! number of time steps, time step [au]
irandom=1651563,        ! random seed
irest=0,                ! should we restart from restart.xyz?

nwrite=1,               ! how often some output should be printed (energies etc.)
nwritex=5,              ! how often should we print coordinates?
nrest=5,                ! how often should we print restart files?
nwritev=0,              ! how often should we print velocities?
/  ! end of a namelist

&nhcopt
temp=298.15,            ! Temperature [K] for Maxwell-Boltzmann sampling and thermostat.
inose=1,                ! Thermostatting: 0=no thermostat, 1=Nose-Hoover 1,2=GLE
tau0=0.001,             ! relaxation time of the NHC thermostat
/
```

Besides these two input files, you need to prepare the directory with a bash script calling the specified *ab initio* program e. g. G09/r.g09. You have to modify this file and specify the method, basis set and other parameters specifying the ab initio calculation. The interface can be found in the folder INTERFACE.

After you prepared your input files, you can run ABIN simply as:

$$\$ \quad bin/abin.v1.0 \quad [-c\ coordinates.xyz]\ [-i\ input.dat] \quad > output$$

Without the command line options, the program looks for initial coordinates in file *mini.dat* and input parameters in *input.in*.

If you run ABIN in cluster environment, it is better to use the automated scripts as described in the section 2.3.

2.2 Output files

At the beginning of the simulation, ABIN dumps all the input parameters (including the defaults not specified by the user) to the standard output along with some other essential information like atom masses, number of threads in parallel simulations etc. During the simulation, only the timestep and time in femtosecond is printed to standards output, all the other information are printed to separate files.

All output files are simple text files. Besides the files containing the geometries, they should all be readable by Xmgrace and you can thus visualize the time evolution of each quantity very easily. Each file contains a one-line header explaining the quantity in each column.

temper.dat This file contains instantaneous and average temperature. The third column shows the conserved quantity of the respective thermostat (see section about NH ther-

mostat). You should always check, whether this quantity does not drift. Otherwise, the thermostat is probably not working properly.

energies.dat This file contains various types of energies. The content may vary slightly depending on the type of simulation. However, the columns are usually in the following order: *time*, *potential energy*, *kinetic energy*, *total energy*.

movie_mini.xyz Contains trajectory from the equilibration period (see parameter *imini*). Thus it is usually not used for subsequent analysis. Note however, that all previous files contains data from equilibration period.

movie.xyz This file contains the trajectory in the xyz format and is readily readable e.g. by VMD or Molden.

restart.xyz This file contains all the necessary information so that ABIN can restart the simulation in case it crashed or if the user wants to extend it. All data are given in atomic units. The structure of this file depends on the simulation parameters (*e.g.* which thermostat you use). The first line contains the current time step. Then there is a section containing the current geometry followed by the section containing the velocities. The third section contains either the thermostat parameters (if you use one) or surface hopping wave function. The next section contains the cumulative averages of certain estimators. The last part contains the state of random number generator. This part is not mandatory. If it's missing, the random number generator simply uses the random seed from the input file and starts the sequence from the beginning.

If you need to produce restart file manually (i.e. for simulation starting with predetermined initial velocities) you must always use the correct character string that comes at the beginning of each section (i.e. "*Cartesian Velocities [au]*"). For surface hopping or classical MD, you can use program UTILS/HARMONIC_SAMPLING/make_restart.f90 to produce the appropriate restart file.

The other output files are present only when you using certain methods. These are described later in the manual.

2.3 Running Abin on clusters

There are a few scripts that are recommended for efficient execution of ABIN jobs on clusters. All of them are simple bash scripts and it should be straightforward to customize them to your needs.

The launching script r.abin (it should be located at the same directory as your input files) can be submitted simply as follows:

$ *qsub -cwd r.abin* (you may also want to specify the queue by *-q QueueName*)

The script copies all files in the current working directory to the directory on the node scratch and launches ABIN there to avoid I/O traffic over the network. The names of the node and the scratch directory are written to file job.log

In order to see the current data from the run, one can use the script fetchabin.sh simply as:

$ $fetchabin.sh$ (assuming it is in your $PATH$).

This will copy the files from the scratch to the current working directory.
To interrupt the simulation, use:

```
$ ExitAbin.sh
```

This script will create empty file *EXIT* in the scratch directory. ABIN checks for the existence of this file at the beginning of each step and if present, prints the restart file and exits. There are two advantages of this approach over simple *qdel* command. First, you have the most recent restart file. Moreover, the files from scratch disk are automatically deleted (if you used script r.abin).

2.4 Running classical MD

Here we will see how to setup a basic classical NVT simulation using velocity Verlet algorithm to propagate classical Newton's equations and Nosé-Hoover-Chain thermostat to keep constant temperature. It is supposed that you have a clean directory with the xyz geometry of your system.

- The first thing to decide is what program to use to get forces and energies. If you would choose for example ORCA, copy the directory ABIN-INTERFACES/ORCA. Now you have to modify script ORCA/r.orca and specify the method and basis that you want to use. The job of this script is simple: take the geometry from ABIN , create input for ORCA, launch ORCA, collect data from ORCA's output and send them back to ABIN .

- Now you need to create or modify the file with input parameters. Copy the sample file from SAMPLES/input.in.cmd. This file is commented so it should be straightforward to understand. There are a few basic things that need to be specified in every simulation: where do we get forces (parameter *pot*), how many atoms we have (*natom*), what type of simulation do we want (*ipimd*, how many time steps and their length (*nstep, dt*). These keywords are in namelist "general". There are two more critical parameters in namelist "nhcopt". You need to specify the thermostat (in this case we will use Nosé-Hoover-Chain thermostat, inose=1) and temperature of the system (*temp*). The keyword *temp* also specifies the initial temperature i.e. the Maxvell-Boltzmann distribution from which we randomly take initial velocities. And that's basically it.

- Now you can launch the simulation. If you are using the cluster environment, copy the script UTILS/r.abin. You should modify it and specify the names of your input files and and path to the ABIN binary (please, copy the binary somewhere to your home folder i.e. $\sim$/bin. Then you can send the job to queue and use fetchabin.sh script too see, what's happening. It is strongly advised that each simulation has its own folder and that there is no other stuff in that folder. Otherwise, fetchabin.sh will copy unnecessary amount of data and will rewrite any changes that you could make since the simulation began.

There are couple of other options that you typically want to set. Each simulation should have its unique random seed (*irandom*). This is especially critical for surface hopping simulations and simulations utilizing quantum thermostat ($inose = 2$). Secondly, you might want to control the frequency of writing to the output to files (*nwrite, nwritex, nrest, nwritev*). Especially the trajectory file movie.xyz might get very big for large systems and/or long simulations. If you are doing thermodynamical simulations ($inose > 0$) you don't need every geometry anyway, since they are correlated and therefore useless for statistical analysis.

When running a classical simulation without thermostat (*inose=0*) you should always check the conservation of energy in file energies.dat. If you use thermostat ($inose > 0$), you should check the conserved quantity of the thermostat, which is printed at the fourth column in file temper.dat. This might vividly oscillate, but it should not drift. If it drifts, then your time step is too long, or you need to tune the parameters of the thermostat, as explained below the next section.

2.4.1 Tuning Nosé-Hoover thermostat

A comprehensive review of Nosé-Hoover algorithm is given in section 5.2. Here we describe only practical details.

It is sometimes necessary to tune the parameters of the thermostat if its conserved quantity drifts or if it even produces incorrect temperature. There are couple of parameters that are needed to specify the action of the thermostat. All of them have default values, which were chosen for typical systems containing hydrogen atoms. The most important parameter is so called relaxation time of the thermostat ($tau0$) given in picoseconds. If this value is not chosen appropriately, the thermostat may decouple from the system and will not work. If your system does not contain fast vibrations, you may want to set a larger value.

Other parameters are more robust. Number of chains ($nchain = 4$) and number of inner iterations during of the NHC integrator ($nrespnose$, $nyosh$) have by default high values that should suffice for common usage. Since the simulation time is typically determined by *ab initio* calculations, the overhead caused by these conservative settings is usually completely negligible.

ABIN by default uses so called massive thermostatting scheme originally devised for PIMD simulations ($imasst = 1$). If you know what you are doing, you might want to use global thermostat, but be aware that this is not tested nearly as much as the massive version and can in some cases provide wrong results for small systems, because ABIN does not remove initial rotations. The details about the global version can be found in section 2.8 which explains how to set up the simulation with SHAKE constraints.

2.5 Restarting the simulation

Quite often you will need to restart the simulation from the last point. This is typically pretty straightforward. Simply set *irest=1* in namelist "general" and start the simulation again.

Note that ABIN always checks for the presence of files movie.xyz and restart.xyz. If they are present but *irest=0*, the program will stop with an error. This prevents an inadvertent deletion of your precious data. However, if you delete movie.xyz and restart.xyz and set *irest=0*, all other output files will be rewritten.

Before reading any data, ABIN copies the original restart.xyz file to "restart.xyz.*timestep*". This file serves as a backup if anything goes wrong, so that you can always go back to that restart file.

WARNING: If the frequency of restart printing ($nrest$) is different from the other ones ($nwrite$, $nwritex$), it may happen that there will be duplicate entries in the output files after the restart. However, you can easily fix this after the end of simulation by simply deleting the conflicting lines and geometries. You can easily determine, whether the simulation was restarted by searching for headers in the output files (lines beginning with "# Time").

In same rare cases, you may want to change the parameters of the simulations when before restart. This may cause troubles because the format might be different. If you want for example to turn on the thermostat, the restart file will be missing the thermostat section. You may fix that by setting *readNHC=0* or *readQT=0*. In other more complicated cases, you will have to manually edit the restart file. As mentioned before, ABIN is strict about the exact string match in order to prevent reading corrupted restart file.

If you want to see what's really going on during restart, checkout the subroutine *restin* in src/analysis.f90. If you want to create the restart file for surface hopping simulation, you should use/modify program UTILS/ABIN-MULTIPLERUNS/make_restart.f90.

2.6 Running Path Integral MD

TODO:

PIMD simulation produces one more output file – *est_energy.dat* – which contains the instantaneous values and cumulative averages of primitive and centroid virial total energy estimators (equations 8 a 9). The cumulative average of the virial estimator should converge much more quickly to an constant value. However, both estimators should always converge to the same value. If they do not, something is wrong with your simulation.

Note that during PIMD simulations ABIN always prints geometries of every bead.

2.6.1 Running parallel simulations

PIMD algorithm can be straightforwardly parallelized, as we need to do do *nwalk* independent *ab initio* calculations at each time step. ABIN uses OpenMP parallelization for this. Note that this is the only part of the code, that is currently parallelized.

You can set variable *nproc* to specify the number of processors you want to use. As you might suspect, *nwalk* should be divisible by *nproc* for maximum efficiency (ABIN will complain if it is not).

Note that you can in addition use parallel version of the *ab initio* program as well. You can of course do that for all types of simulations. However, in the case of PIMD, it is almost always more efficient to use ABIN parallelization, since you rarely get perfect linear scaling in *ab initio* codes.

For example, it might be convenient to set *nproc=4* when using PI-GLE method with 4 beads (see below), and also set 2 processors for each ab initio calculation. This setup requires 8 processors in total.

2.7 Running simulations with GLE thermostat

Besides the Nosé-Hoover thermostat, there is a thermostat based on Generalized Langevin equation. While this thermostat can be used also for classical MD and has many different application, it is currently used to simulate quantum nuclear effects using either Quantum Thermostat or PI+GLE method as described below.

2.8 Quantum thermostat

The quantum thermostat method (here denoted QT) was devised by M. Cerriotti et al. for efficient simulations of quantum nuclear effects without the need for Path Integrals, which are very computationally demanding. There are 2 key ideas behind this method:

- The QT method is based on the solution of harmonic oscillator in thermal equilibrium. It turns out that both classical and quantum distributions have Gaussian shape, albeit with different widths. One can then convert the classical distribution into the quantum distribution by classical MD at elevated temperature $T^{\star}$ given by:

$$= \tag{1}$$

2.9 Running Surface Hopping simulations

TODO:

2.10 SHAKE and global NH thermostat

TODO:

imasst=0 natmolt=2

2.11 Boundary conditions

By default, all simulations are performed under open boundary conditions. For classical simulation with Amber force fields (keyword pot=nab), the periodic boundary conditions (**PBC**) are being developed. For cluster simulations, one can use simple spherical restraining potential (**SBC**) as explained in detail below.

2.11.1 SBC

TODO

3 List of keywords

This section contains a rather extensive list of all the keywords. Most of the keywords have default values, which are given in the 2nd column. Some of the keywords may be rather obscure and related only to technicalities or special applications. If you feel that you do not understand some keyword, don't worry, it's probably not important. The keywords that you most likely don't need to care about are given with grey background. Note that the program will sometimes prevent you from using experimental features, weird keyword combinations or debug options unless you set $iknow = 1$ in namelist "general". It also tries to sanitize user input against obvious errors. But do not rely on this feature! You can still make your simulation meaningless by bad simulation parameters.

The keywords are divided into namelists that they belong to. The namelists *general* and *nhc* must always be present in the input file. Order of namelists in the input file is irrelevant.

3.1 &general

keyword	**default**	Description
pot	-	Specifies, where do we get forces and energies. Supported options are: **g09**, **orca**, **tera**, **turbo**, **molpro** **dyn** (user specified script ./DYN/.r.dyn) **nab** (amber force field), **harm** (diatomic harmonic oscillator), **morse** (morse potential for diatomic molecule), **guillot**[CITE], **2dho** (2-dimensional one-atom harmonic oscillator)
natom	–	Number of atoms.
pimd	**0**	Controls the type of simulation. **0** – Classical simulation **1** – Path-integral simulation **2** – Surface Hopping **3** – Simple steepest-descent minimization
nstep	**1**	Number of time steps.
dt	**20**	Time step [a. u.]
irandom	**156873**	Random seed
irest	**0**	Controls the restart of the simulation. **0** – Do not restart. Take the initial structure from mini.dat If restart.xyz is present, end with and error for safety reasons. **1** – Restart the simulation from file "restart.xyz". Copy the file "restart.xyz" to "restart.xyz.*it*", where *it* is the current timestep.
imini	**0**	Number of time steps for equilibration of the system. During this period, the coord
conatom	**0**	Number of frozen atoms.
		Keywords for PIMD and PI-GLE simulations
nwalk	**1**	Number of random walkers (beads) for PIMD. It is automatically set to 1 if *pimd=0*.
istage	**0**	Controls the coordinates used for PIMD. **0** – Normal cartesian coordinates. **1** – Staging coordinates; these should we always used with PIMD.
nproc	**1**	Number of processors for parallel simulations. Should be 1 or equal to *nwalk*.
nabin	**50**	Number of substeps in RESPA algorithm used for PIMD.
		Options controlling the output
nwrite	**1**	Print basic output every *nwrite* step (e.g. temperature, energies).
nwritex	**1**	Print coordinates every *nwritex* step.
nrest	**1**	Print restart file every *nrest* step. Larger value is recomended, since restart files can be quite big.
nwritev	**0**	Print velocities every *nwritev* step.
ncalc	**1**	Calculate observables (energies, temperature) every *ncalc* step.

		OTHER OPTIONS
isbc	**0**	Controls spherical boundary conditions (SBC). **0** – Off. **1** – On.
rb_sbc	–	Radius of the sphere for SBC in Angstroms. If not given, the radius of the initial structure is used.
kb_sbc	**0.02**	Force constant for SBC in atomic units.
anal_ext	**0**	Should we call external analysis routine? User can write his/her own routine, which is called during analysis. Template for this subroutine is in file analyze_ext_template.f90.
icv	**0**	Should we calculate constant volume heat capacities?
ihess	**0**	Should we compute hessian for heat capacities? Currently only implemented for pot=*nab*,*harm* or *morse*.

3.2 &system

keyword	**default**	Description
massnames	*array*	Array of atom names, for which we define user-specified masses.
masses	*array*	Array of user-specified masses. This array should match the *massnames* array.
		SHAKE OPTIONS
nshake	**0**	Number of fixed bonds.
shake_tol	**0.001**	Shake convergence tolerance
ishake1	*array*	Array of first-atom indices specifying the fixed bonds.
ishake2	*array*	Array of second-atom indices specifying the fixed bonds.
		OPTIONS RELATED TO GEOMETRICAL ANALYSES
ndist	**0**	Number of distances for analysis.
nang	**0**	Number of angles for analysis.
ndih	**0**	Number of dihedral angles for analysis.
dist1	*array*	Array of first-atom indices for bond definition.
dist2	*array*	Array of second-atom indices for bond definition.
ang1	*array*	Array of first-atom indices for angle definition.
xmin	0.5	Minimum bond distance for binning.
xmax	5.0	Maximum bond distance for binning.
disterror	**0**	If set to **1**, abort the simulation if the bond distance is outside the range (*xmin*,*xmax*).
ang2	*array*	Array of second-atom indices for angle definition.
ang3	*array*	Array of third-atom indices for angle definition.
dih1	*array*	Array of first-atom indices for dihedral definition.
dih2	*array*	Array of second-atom indices for dihedral definition.
dih3	*array*	Array of third-atom indices for dihedral definition.
dih4	*array*	Array of fourth-atom indices for dihedral definition.
nbin	**2000**	Number of bins for the histograms of bond densities.
nbin_ang	**180**	Number of bins for the histograms of angles.

3.3 &nhcopt

keyword	**default**	Description
temp	**0.0**	Temperature in Kelvins. This also controls the generation of initial velocities if *temp0* is not explicitly set.
temp0	–	Temperature of initial Maxvell-Boltzmann distribution.
inose	**0**	Controls the use of a thermostat. **0** – Microcanonical trajectory. No thermostat. **1** – Nosé-Hoover thermostat. **2** – Quantum GLE thermostat.
nchain	**4**	Number of Nosé-Hoover chains. This should be always > 1 for small systems.
tau0	**0.001**	Relaxation time of the thermostat in picoseconds. From this parameter, the thermostat masses are determined automatically based on temperature and size of the system.[14]
imasst	**1**	Determines, whether we use massive or global thermostat. **0** – Use global thermostat. Requires further specification of the system using *nmolt*, *natmolt* and *nshakemol* variables. WARNING: This was not thoroughly tested. Since we do not remove rotations, this thermostat can produce wrong results. It is however required if you use SHAKE. **1** – Use massive thermostat. Each degree of freedom has its own NH chain. This is required for PIMD and recommended for classical simulations.
nmolt	**1**	If we use global NHC thermostat, the system is divided into *nmolt* parts and each part has its own NHC thermostat.
natmolt	*array*	Array of integers specifying the number of atoms in each part that has its own thermostat. This assumes, that the parts are consecutively aligned in the input coordinates.
nshakemol	**0**	Array of integers specifying a number of fixed bonds in each part of the system that has its own thermostat. Note that you can not fix bonds between atoms from different parts of the system.
nrespnose	**3**	Number of sub-steps of the NHC integrator.[14]
nyosh	**7**	Number of weights in the Suzuki-Yoshida scheme.[14] **1** - Do not use Suzuki-Yoshida scheme (not recommended) 3 – 4th order Suzuki-Yoshida scheme. 7 – 6th order Suzuki-Yoshida scheme.
ams	–	Nosé-Hoover mass. This number is always chosen automatically for PIMD simulations. This parameter overrides the *tau0* parameter.
scaleveloc	**1**	Whether we scale initial velocities so that the total kinetic energy matches the desired temperature. This should reduce the initial oscillations of the thermostat.
initnhc	**1**	Whether we initialize NHC momenta.
readnhc	**1**	Read NHC momenta from restart.xyz if $inose = 1$.

3.4 &sh

This namelist contains parameters for surface hopping simulations and is read only if *ipimd*=3.

keyword	**default**	Description
nstate	**1**	Number of electronic adiabatic states.
istate_init	**1**	Initial electronic state.
deltae	**5.0**	Energy threshold for the computation of NAC vectors. The NAC vector between states is not computed if the difference in energy is $>$ *deltae*.
popthr	**0.001**	The NAC vector between two states is not computed, unless at least one of them has population $>$ *popthr*. Set this to some negative number to disable this feature.
nac_accu1	**7**	Default accuracy of NAC vector is 10^{-nac_accu1}.
nac_accu2	**5**	If the calculation of NAC does not converge, try with accuracy 10^{-nac_accu2}.
energydifthr	**1.0**	Abort the simulation if the total energy in two consecutive time steps differs more than *energydifthr*.
energydriftthr	**1.0**	Abort the simulation the total energy drifts from its initial value more than this threshold. This criterion is more strict than *energydifthr*.
popsumthr	**0.001**	If the norm of the electronic wave function differs from 1 more than *popthr*, the simulation is aborted. You should probably increase *substep*.
alpha	**0.1**	Empirical parameter for the decoherence correction.[CITE] If you do not want to use decoherence correction, set *alpha* $<= 0$.
substep	**100**	Number of substeps for the integration of electronic SE.
integ	**butcher**	Integrator of the electronic Schrödinger equation. **euler** – Euler integrator. Only for debugging purposes! **rk4** – Runge-Kutta 4th-order integrator. Not recommended. **butcher** – Butcher's 5th-order integrator. Default and recommended.
nohop	**0**	Hopping is forbidden if *nohop* = 1.
inac	**0**	Controls whether we use NAC couplings or time-derivative couplings. **0** – Use non-adiabatic couplings from *ab initio* program. **1** – Use time-derivative couplings.
adjmom	**0**	Controls the adjustment of velocity after the hopping event. **0** – Adjust the velocity along the direction of NAC vector. If there is not enough kinetic energy in that direction, try simple scaling as if *adjmom*=1. **1** – Scale the velocity, if there is enough kinetic energy. This is the only and default option if *inac*=1.
revmom	**0**	Should we reverse the velocity in case of frustrated hop? **0** – No. **1** – Yes.
phase	**0**	Should we integrate the phase along the trajectory? **0** – No. **1** – Yes.

4 Interface with electronic structure codes

5 Detailed descriptions of the algorithms

5.1 PIMD with staging transformation

There are three essential aspects of an efficient PIMD scheme: Coordinate transformation that uncouples the quantum harmonic terms (either staging or normal modes), efficient thermostatting and multiple time step algorithm. These will be all described in detail in next few paragraphs, including working equations as implemented in the ABIN code.

Staging transformation is defined by the recursive relations

$$\begin{aligned} q_1 &= x_1 \\ q_k &= x_k - \frac{(k-1)x_{k+1} + x_1}{k}, k = 2, \ldots, P, \end{aligned} \tag{2}$$

with the inverse transformation:

$$\begin{aligned} x_1 &= q_1 \\ x_k &= q_k + \frac{k-1}{k} x_{k+1} + \frac{1}{k} q_1, k = 2, \ldots, P. \end{aligned} \tag{3}$$

Similar relations hold for the transformation of forces. The effective Hamiltonian of an extended system with P beads then transforms to:

$$H_{\text{eff}} = \sum_{k=1}^{P} \left[\frac{p_k^2}{2m_k^*} + \frac{1}{2} m_k \omega_P^2 q_k^2 + \frac{1}{P} U(q_k) \right], \tag{4}$$

where $\omega_P = \sqrt{P}/\beta\hbar$ and parameters m_k are defined as:

$$m_1 = 0, \tag{5}$$

$$m_k = \frac{k}{k-1} m, k = 2, \ldots, P, \tag{6}$$

and fictitious masses m_k^* are chosen as $m_1^* = m$ and $m_k^* = m_k$. The Hamiltonian is further supplemented by Nosé-Hoover chain of length M (to be discussed in the next Section) so that the final set of equations of motion read as follows:

$$\dot{q}_k = \frac{p_k}{m_k^*}, \tag{7a}$$

$$\dot{p}_k = -m_k \omega_P^2 q_k - \frac{1}{P} \frac{\partial U}{\partial q_k} - \frac{p_{\eta_{k,1}}}{Q_1} p_k, \tag{7b}$$

$$\dot{\eta}_{k,\gamma} = \frac{p_{\eta_{k,\gamma}}}{Q_k}, \tag{7c}$$

$$\dot{p}_{\eta_{k,1}} = \frac{p_k^2}{m_k} - kT - \frac{p_{\eta_{k,2}}}{Q_k} p_{\eta_{k,1}}, \tag{7d}$$

$$\dot{p}_{\eta_{k,1}} = \left[\frac{p_{\eta_{k,\gamma-1}}^2}{m_k} - kT \right] - \frac{p_{\eta_{k,\gamma+1}}}{Q_k} p_{\eta_{k,\gamma}} \gamma = 2, \ldots, M-1, \tag{7e}$$

$$\dot{p}_{\eta_{k,M}} = \left[\frac{p_{\eta_{k,M-1}}^2}{Q_k} - kT \right]. \tag{7f}$$

The thermostat masses are set to $Q_k = kT/\omega_P^2$. The mass corresponding to the first bead, which happens to be the centroid of the bead necklace, can alternatively be set to $Q_1 = kT\tau^2$, where τ s the characteristic time scale of the system.

The conserved quantity of these equations is:

$$H = \frac{p_k^2}{2m_k^*} + U + \sum_{\gamma=1}^{M}\left[\frac{p_{\eta_\gamma}^2}{2Q_k} + kT\eta_\gamma\right]. \tag{8}$$

This quantity should not drift over time as we are using a time-reversible integrator (see below). One should also always check the convergence of the primitive and centroid-virial energy estimators (Eqs. 8 and 9) as these should always converge to the same value regardless of number of beads. Failure to do so usually indicates that the time step is too long.

$$\epsilon_{\rm P} = \frac{3NP}{2\beta} - \frac{1}{2}\omega_P^2\sum_{i=1}^{N} m_i \sum_{s=1}^{P}|\mathbf{r}_{i,s} - \mathbf{r}_{i,s+1}|^2 + \overline{U}, \tag{9}$$

$$\epsilon_{\rm CV} = \frac{3N}{2\beta} + \frac{1}{2P}\sum_{i=1}^{N}\sum_{s=1}^{P}(\mathbf{r}_{i,s} - \mathbf{r}_i^{(c)})\cdot\left(\frac{\partial\overline{U}}{\partial\mathbf{r}_{i,s}}\right) + \overline{U}, \tag{10}$$

where $\overline{U} = \frac{1}{P}\sum_{s=1}^{P} U(\mathbf{r}_{1,s},\ldots,\mathbf{r}_{N,s})$ and $\mathbf{r}_i^{(c)} = \sum_{s=1}^{P} r_{i,s}/P$ are centroid variables.

Eqs. 6 are integrated using RESPA multiple time step algorithm according to he following splitting of the Liouville operator:

$$iL = iL_{\rm quant} + iL_{\rm kin} + iL_{\rm clas} + iL_{\rm NHC}, \tag{11}$$

where the operators for kinetic, quantum and classical parts are defined as:

$$iL_{\rm kin} = \sum_{k=1}^{P}\frac{p_k}{m_k^*}\frac{\partial}{\partial q_k}, \tag{12}$$

$$iL_{\rm quant} = -\sum_{k=1}^{P} m_k\omega_P^2 q_k\frac{\partial}{\partial p_k}, \tag{13}$$

$$iL_{\rm clas} = -\sum_{k=1}^{P}\frac{1}{P}\frac{\partial U}{\partial q_k}\frac{\partial}{\partial p_k}. \tag{14}$$

The operator for Nosé-Hoover part $iL_{\rm NHC}$ will be discussed separately in the next Section. The propagator $e^{iL\Delta t}$ can be factorized into:

$$\begin{aligned} e^{iL\Delta t} = {} & e^{iL_{\rm NHC}\delta t/2}e^{iL_{\rm clas}\Delta t/2}e^{-iL_{\rm NHC}\delta t/2} \\ & \left[e^{iL_{\rm NHC}\delta t/2}e^{iL_{\rm quant}\delta t/2}e^{iL_{\rm kin}\Delta t}e^{iL_{\rm quant}\delta t/2}e^{iL_{\rm NHC}\delta t/2}\right]^n \\ & e^{iL_{\rm NHC}\delta t/2}e^{iL_{\rm clas}\Delta t/2}e^{iL_{\rm NHC}\delta t/2}. \end{aligned} \tag{15}$$

Note that the propagator $e^{iL_{\rm NHC}\delta t/2}$ is applied in the inner loop in order to efficiently thermostat the rapidly varying harmonic motion (so called XI-RESPA scheme). In this scheme, the quantum forces are evaluated at n-times shorter time step. The classical forces and energies need to be calculated only once per Δt, which is very important mainly for *ab initio* PIMD, where the calculation of classical forces is very expensive. Note that simple velocity Verlet algorithm sandwiched by $e^{iL_{\rm NHC}\Delta t/2}$ is recovered by setting $n = 1$.

Eq. 14 can be transformed into the computer algorithm by noticing that all the operators (except for Nosé-Hoover part) are of the form $e^{c\frac{\partial}{\partial x}}$. It can be shown, that these operators represent simple translation[14]:

$$e^{\frac{p_k}{m_k}\frac{\partial}{\partial q_k}\Delta t}\, q_k = q_k + \frac{p}{m}\Delta t.$$

5.2 Integration of Nosé-Hoover Chain equations

There are currently two versions of the code for propagating the NHC equations in the ABIN program. The massive version attach NHC chain to each degree of freedom and is primarily used for the PIMD simulations, while the second versions is more general and allows the user to specify the subsystems that are coupled separately to the thermostat. One chain is usually attached to the whole system in case of CMD. However, in this global version, only total kinetic energy is controlled so hot and cold spots can theoretically develop in the system. This version is necessary for simulations with constraints, since it does not break the constraint conditions. For example, if we want to simulate rigid water model, we can couple each water molecule to a separate thermostat, but we cannot thermostat each atom, since that would break the constraints.

The equations for the global version of NHC thermostat are only slightly different from the massive version described above. One simply replace Eq. 6d by:

$$\dot{p}_{\eta_1} = \left[\sum_{i=1}^{N} \frac{\mathbf{p}_i}{m_i} - 3NkT\right] - \frac{p_{\eta_1}}{Q_1}\mathbf{p}_i, \tag{16}$$

where N is number of thermostatted particles. The conserved quantity is now:

$$H = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} + U + 3NkT\eta_1 + \sum_{\gamma=1}^{M}\left[\frac{p_{\eta_\gamma}}{2Q_\gamma}\right] + \sum_{\gamma=2}^{M} kT\eta_\gamma. \tag{17}$$

Note that the indexing of the masses Q is now different.The masses in the massive PIMD scheme may be different for each bead, but they are the same for each index γ. In this case, they may differ for each index γ. Namely, it can be shown that optimal masses are[1]:

$$\begin{aligned} Q_1 &= 3NkT\tau^2, \\ Q_\gamma &= kT\tau^2\gamma = 2, \ldots, M, \end{aligned} \tag{18}$$

where τ is the characteristic time scale of the system. It is convenient to set the thermostat masses using this parameter, because the optimal values of the masses themselves depend on system size and target temperature. If the masses are set inappropriately, the thermostat may decouple from the system and the correct temperature will not be obtained.

The integration of NHC equations is generally difficult, since the right hand side depends on velocities. I implemented the approach of Tuckerman *et al.* who used the Liouville operator formalism to derive the time-reversible algorithm for propagation[15]. Here I describe the implementation of the global version. The implementation of the massive version is completely analogous.

The Liouville operator for NHC is:

$$\begin{aligned} iL_{\mathrm{NHC}} &= -\sum_{i=1}^{N} \frac{p_{\eta_1}}{Q_1}\mathbf{p}_i \cdot \frac{\partial}{\partial \mathbf{p}_i} + \sum_{\gamma=1}^{M} \frac{p_{\eta_\gamma}}{Q_\gamma}\frac{\partial}{\partial \eta_\gamma} \\ &+ \sum_{\gamma=1}^{M-1}\left(G_\gamma - p_{\eta_\gamma}\frac{p_{\eta_{\gamma+1}}}{Q_{\gamma+1}}\right)\frac{\partial}{\partial p_{\eta_\gamma}} + G_M\frac{\partial}{\partial p_{\eta_M}}, \end{aligned} \tag{19}$$

where the thermostat "forces" are defined as:

$$\begin{aligned} G_1 &= \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{m_i} - 3NkT, \\ G_\gamma &= \frac{p_{\eta_{\gamma-1}}}{Q_{\gamma-1}} - kT. \end{aligned} \tag{20}$$

The primitive factorization of the propagator $e^{iL_{\mathrm{NHC}}\delta t/2}$ was done as:

$$
\begin{aligned}
S(\delta/2) = \exp\left[\frac{\delta}{4}G_M\frac{\partial}{\partial p_{\eta_M}}\right] \\
\times \prod_{\gamma=M-1}^{1}\left\{\exp\left[-\frac{\delta}{8}\frac{p_{\eta_{\gamma+1}}}{Q_{\gamma+1}}p_{\eta_\gamma}\frac{\partial}{\partial p_{\eta_\gamma}}\right]\exp\left[\frac{\delta}{4}G_\gamma\frac{\partial}{\partial p_{\eta_\gamma}}\right]\exp\left[-\frac{\delta}{8}\frac{p_{\eta_{\gamma+1}}}{Q_{\gamma+1}}p_{\eta_\gamma}\frac{\partial}{\partial p_{\eta_\gamma}}\right]\right\} \\
\times\prod_{i=1}^{N}\exp\left[-\frac{\delta}{2}\frac{p_{\eta_1}}{Q_1}\mathbf{p}_i\cdot\frac{\partial}{\partial\mathbf{p}_i}\right]\prod_{\gamma=1}^{M}\exp\left[\frac{\delta}{2}\frac{p_{\eta_\gamma}}{Q_\gamma}\frac{\partial}{\partial\eta_\gamma}\right] \\
\times \prod_{\gamma=1}^{M-1}\left\{\exp\left[-\frac{\delta}{8}\frac{p_{\eta_{\gamma+1}}}{Q_{\gamma+1}}p_{\eta_\gamma}\frac{\partial}{\partial p_{\eta_\gamma}}\right]\exp\left[\frac{\delta}{4}G_\gamma\frac{\partial}{\partial p_{\eta_\gamma}}\right]\exp\left[-\frac{\delta}{8}\frac{p_{\eta_{\gamma+1}}}{Q_{\gamma+1}}p_{\eta_\gamma}\frac{\partial}{\partial p_{\eta_\gamma}}\right]\right\} \\
\times\exp\left[\frac{\delta}{4}G_M\frac{\partial}{\partial p_{\eta_M}}\right]. \qquad (21)
\end{aligned}
$$

However, this factorization would severely limit the time step since the thermostat forces are rapidly varying. One can again exploit the RESPA scheme. Still, the number of inner time step n_c would be very high. To reduce this number, I used the Suzuki-Yoshida scheme[16], in which the time step is multiplied by a set of numerical weights, which results in higher order method than the simple Trotter factorization. The final factorization of the NHC propagator can be written as follows:

$$
e^{iL_{\mathrm{NHC}}\Delta t/2} \approx \left[\prod_{\alpha=1}^{n_{\mathrm{sy}}} S(w_\alpha\Delta t/2n)\right]^{n_c}. \qquad (22)
$$

Suzuki-Yoshida weights w_α for fourth-order method and sixth-order method are currently supported.

The primitive factorization in Eq. 20 contains translational operators introduced above and also operators of the form $e^{cx\frac{\partial}{\partial x}}$. These can be shown to act as a scaling operators:

$$
\exp\left[cx\frac{\partial}{\partial x}\right]x = xe^c. \qquad (23)
$$

Therefore, the factorization in Eq. 20 can be also directly translated to the computer code.

5.3 Quantum Thermostat

Unlike the PIMD method, the quantum thermostat (QT) technique is only an approximate way to describe the quantum nuclear effects. The code in ABIN is based on the implementation by Parinello *et al.*[10].

The quantum thermostat is based on Generalized Langevin equation (GLE), which is a stochastic integro-differential equation. The general theory is rather involved, so only the basic ideas are explained here. Interested reader is referred to the original literature[10].

Integro-differential equation that is equations involving both derivatives and integrals are rather difficult to implement as the evolution of the system depends on the entire system history. Fortunately, the GLE can be cast into the simple matrix form using additional dynamical variables as follows (considering only one particle with mass m)[17]:

$$\dot{q} = \frac{p}{m}, \tag{24}$$

$$\begin{pmatrix} \dot{p} \\ \dot{\mathbf{s}} \end{pmatrix} = \begin{pmatrix} -U'(q) \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} a_{pp}\mathbf{a}_p^T \\ \bar{\mathbf{a}}_p\mathbf{A} \end{pmatrix} \begin{pmatrix} p \\ \mathbf{s} \end{pmatrix} + \begin{pmatrix} b_{pp}\mathbf{b}_p^T \\ \mathbf{b}_p\mathbf{B} \end{pmatrix} \begin{pmatrix} \xi \end{pmatrix},$$

where $\mathbf{s}$ is the vector of n additional degree of freedoms, ξ is the vector of $n+1$ uncorrelated Gaussian random numbers. For $n = 0$, the standard Langevin equation is recovered.

$$\dot{q} = p \tag{25}$$

$$\dot{p} = -U'(q) - a_{pp}p + b_{pp}\xi(t),$$

where a_{pp} is the friction coefficient and b_{pp} is the intensity of the random force. These are related through the so-called fluctuation-dissipation theorem (FDT), which reads as:

$$b_{pp}^2 = 2a_{pp}mk_{\mathrm{B}}T. \tag{26}$$

The matrices in the Eq. 23 are related through so-called drift matrix $\mathbf{C}$ as[17]:

$$\mathbf{A}_p\mathbf{C}_p + \mathbf{C}_p\mathbf{A}_p^T = \mathbf{B}_p\mathbf{B}_p^T, \tag{27}$$

with the notation $\mathbf{M}_p = \begin{pmatrix} m_{pp}\mathbf{m}_p^T \\ \bar{\mathbf{m}}_p\mathbf{M} \end{pmatrix}$. By setting $\mathbf{C}_p = k_BT$, the Eq. 23 satisfy the FDT and and thus generates canonical distributions at given temperature.

Eq. 23 can be solved analytically for the harmonic oscillator. Therefore, for a given frequency, we know how the system responds to the thermostat and we can thus optimize the parameters of the matrices to obtain a desired response. For example, the parameters can be optimized for the optimal sampling of classical phase space. One can also maintain different normal modes at different temperatures (frequency dependent thermostatting) and this possibility is the basis for simulating quantum effects.

One can exploit the fact that both quantum and classical distributions have gaussian shape and differ only in the standard deviation. By setting the effective temperature to $T^*(\omega) = \frac{\hbar\omega}{2k_B}\coth\frac{\hbar\omega}{2k_{\mathrm{B}}T}$ we can convert classical gaussian distribution with standard deviation $\sigma_{\mathrm{c}}^2 = \frac{k_{\mathrm{B}}T}{m\omega^2}$ to the quantum distribution with deviation $\sigma_{\mathrm{q}}^2 = \frac{\hbar}{2m\omega}coth\frac{\hbar\omega}{2k_{\mathrm{B}}T}$ using purely classical simulations. However, we need to maintain different normal modes at different effective temperature, which is exactly what GLE makes possible. By doing this, the simulations are non-equilibrium and do not satisfy FDT theorem.

The fitting of parameters in matrices $\mathbf{A}$ and $\mathbf{C}$ (matrix $\mathbf{B}$ is then computed using Eq. 26) is not straightforward and is not described here. Although the derived parameters are strictly applicable only for systems of coupled harmonic oscillators, they can also be used as

an approximate method for realistic molecular systems, which are in general anharmonic, since they can be viewed as a collection of normal modes coupled by anharmonicities. However, one must be cautious when the anharmonic coupling is too strong. In such cases, the energy may flow between different normal modes (*i.e.* from the high to low frequencies) and the simulation is spoiled. This may be viewed as ZPE leakage, which is a common problem in semiclassical approaches.

The main advantage of QT is its computational simplicity. The computational savings as compared with full PIMD simulations are substantial, especially for *ab initio* simulations. Moreover, the sampling efficiency is much greater using QT method, since the PIMD simulations become less ergodic and difficult to converge with large number of beads.

Another advantage of the QT over the simple PIMD scheme presented here is that it provides at least an approximate momentum distributions. This might be potentially useful for the generation of initial conditions for semiclassical photodynamical simulations.

Finally, it is possible to combine the QT with PIMD (here denoted as PI+GLE method)[11] which results in much faster convergence of quantities with the number of beads. Moreover, the ZPE leakage becomes of lesser concern as we increase the number of beads. It was shown[11] and it is also our experience that only four beads are sufficient for fully converged position distribution. When using normal mode transformation, the GLE parameters can be optimized specifically for each normal mode to speed up the convergence of quantum kinetic energy computed with centroid-virial energy estimator[18]. This so called PIGLET method is not yet implemented. Note that neither PI+GLE nor PIGLET methods provide rigorous momentum distribution function, although other approximate methods can still be used[18].

5.3.1 Implementation

The quantum thermostat in ABIN is based on the sample code from the website *http://gle4md.berlios.de/* which is maintained by the original authors. At present, the GLE thermostat is called on top of the classical velocity Verlet algorithm according to the following pseudocode:

$call\ GleStep(\frac{dt}{2})$
$p = p + force \times \frac{dt}{2}$
$q = q + \frac{p}{m}dt$
$call\ GetForces()$
$p = p + force \times \frac{dt}{2}$
$call\ GleStep(\frac{dt}{2})$

The PI+GLE method is also implemented. The GLE thermostat in this case is part of the RESPA algorithm in the same way as the NHC thermostat. Note, that in PI+GLE method, one has to use physical bead masses i. e. staging transformation is not applicable in this case[18]. More efficient implementation would require the use of normal mode transformation. This is now a work in progress.

The parameters for different simulations can be obtained from the same website as mentioned above. To choose the appropriate parameters, one only needs to know the highest frequency present in the system and the temperature of the simulation. The parameters can be distinguished by a number of additional degrees of freedom s and by the fitted frequency range which is given by the ration $x = \frac{\hbar\omega}{k_{\mathrm{B}}T}$. That means that the maximum frequency for given parameters depends on the temperature of the simulation. In the case of PI+GLE method, the parameters are different for different number of beads.

There is a fundamental difference between GLE and PIGLE parameters. The latter were optimized only to give correct position distributions since the momenta do not have physical meaning in the PIMD method. This approach leaves more freedom in the fitting procedure and

therefore, the PIGLE parameters were also optimized for optimal sampling. Therefore, even for simulations with one bead, PIGLE-P1 parameters should be superior to the original GLE parameters if one is not interested in the momentum distribution.

References

[1] Martyna, G. J.; Klein, M. L.; Tuckerman, M. *J. Chem. Phys.* **1992,** *97,* 2635–2643.

[2] Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Mennucci, B.; Petersson, G. A.; Nakatsuji, H.; Caricato, M.; Li, X.; Hratchian, H. P.; Izmaylov, A. F.; Bloino, J.; Zheng, G.; Sonnenberg, J. L.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Montgomery Jr., J. A.; Peralta, J. E.; Ogliaro, F.; Bearpark, M.; Heyd, J. J.; Brothers, E.; Kudin, K. N.; Staroverov, V. N.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Rega, N.; Millam, J. M.; Klene, M.; Knox, J. E.; Cross, J. B.; Bakken, V.; Adamo, C.; Jaramillo, J.; Gomperts, R.; Stratmann, R. E.; Yazyev, O.; Austin, A. J.; Cammi, R.; Pomelli, C.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Zakrzewski, V. G.; Voth, G. A.; Salvador, P.; Dannenberg, J. J.; Dapprich, S.; Daniels, A. D.; Farkas, O.; Foresman, J. B.; Ortiz, J. V.; Cioslowski, J.; Fox, D. J. "Gaussian 09 Revision A.1", 2009.

[3] Werner, H.-J.; Knowles, P. J.; Manby, F. R.; Schütz, M.; Celani, P.; Knizia, G.; Korona, T.; Lindh, R.; Mitrushenkov, A.; Rauhut, G.; Adler, T. B.; Amos, R. D.; Bernhardsson, A.; Berning, A.; Cooper, D. L.; Deegan, M. J. O.; Dobbyn, A. J.; Eckert, F.; Goll, E.; Hampel, C.; Hesselmann, A.; Hetzer, G.; Hrenar, T.; Jansen, G.; Köppl, C.; Liu, Y.; Lloyd, A. W.; Mata, R. A.; May, A. J.; McNicholas, S. J.; Meyer, W.; Mura, M. E.; Nicklass, A.; Palmieri, P.; Pflüger, K.; Pitzer, R.; Reiher, M.; Shiozaki, T.; Stoll, H.; Stone, A. J.; Tarroni, R.; Thorsteinsson, T.; Wang, M.; Wolf, A.; Others, "MOLPRO, version 2006.1, a package of ab initio programs", 2006.

[4] Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. *J. Comput. Chem.* **1993,** *14,* 1347–1363.

[5] "TURBOMOLE V6.4 2012, a development of University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989-2007, TURBOMOLE GmbH, since 2007; available from http://www.turbomole.com", .

[6] Shao, Y.; Molnar, L. F.; Jung, Y.; Kussmann, J.; Ochsenfeld, C.; Brown, S. T.; Gilbert, A. T. B.; Slipchenko, L. V.; Levchenko, S. V.; O'Neill, D. P.; DiStasio, R. A.; Lochan, R. C.; Wang, T.; Beran, G. J. O.; Besley, N. A.; Herbert, J. M.; Lin, C. Y.; Van Voorhis, T.; Chien, S. H.; Sodt, A.; Steele, R. P.; Rassolov, V. A.; Maslen, P. E.; Korambath, P. P.; Adamson, R. D.; Austin, B.; Baker, J.; Byrd, E. F. C.; Dachsel, H.; Doerksen, R. J.; Dreuw, A.; Dunietz, B. D.; Dutoi, A. D.; Furlani, T. R.; Gwaltney, S. R.; Heyden, A.; Hirata, S.; Hsu, C.-P.; Kedziora, G.; Khalliulin, R. Z.; Klunzinger, P.; Lee, A. M.; Lee, M. S.; Liang, W.; Lotan, I.; Nair, N.; Peters, B.; Proynov, E. I.; Pieniazek, P. A.; Rhee, Y. M.; Ritchie, J.; Rosta, E.; Sherrill, C. D.; Simmonett, A. C.; Subotnik, J. E.; Woodcock, H. L.; Zhang, W.; Bell, A. T.; Chakraborty, A. K.; Chipman, D. M.; Keil, F. J.; Warshel, A.; Hehre, W. J.; Schaefer, H. F.; Kong, J.; Krylov, A. I.; Gill, P. M. W.; Head-Gordon, M. *Phys. Chem. Chem. Phys.* **2006,** *8,* 3172–91.

[7] Neese, F. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2012,** *2,* 73-78.

[8] Ufimtsev, I. S.; Martínez, T. J. *J. Chem. Theory Comput.* **2009,** *5,* 2619–2628.

[9] Barbatti, M. *WIREs Comput. Mol. Sci.* **2011,** *1,* 620–633.

[10] Ceriotti, M. *A novel framework for enhanced molecular dynamics based on the generalized Langevin equation Molecular dynamics simulations Simulation of the trajectories of a system at the atomic level,* Thesis, ETH Zürich, 2010.

[11] Ceriotti, M.; Manolopoulos, D. E.; Parrinello, M. *J. Chem. Phys.* **2011,** *134,* 84104.

[12] "http://www.fftw.org/", .

[13] Li, Y.; Krilov, G.; Berne, B. *J. Phys. Chem. B* **2006,** *110,* 13256-13263.

[14] Tuckerman, M. E. *Statistical Mechanics: Theory and Molecular Simulations;* Oxford University Press: Great Clarendon Street, Oxford, 2010.

[15] Martyna, G. J.; Tuckerman, M. E.; Tobias, D. J.; Klein, M. L. *Mol. Phys.* **1996,** *87,* 1117–1157.

[16] Yoshida, H. *Phys. Lett. A* **1990,** *150,* 262–268.

[17] Ceriotti, M.; Bussi, G.; Parrinello, M. *J. Chem. Theory Comput.* **2010,** *6,* 1170–1180.

[18] Ceriotti, M.; Manolopoulos, D. E. *Phys. Rev. Lett.* **2012,** *109,* 100604.