Born-Oppenheimer Molecular Dynamics in ONETEP

Simon M.-M. Dubois[1] and Valerio Vitale[2]

[1]Cavendish Laboratory, University of Cambridge
[2]School of Chemistry, University of Southampton

March 3, 2016

This document is intended as a guide to the molecular dynamics (MD) functionality in ONETEP (v4.4) [1]. Though some theoretical concepts are reviewed, it is not meant to be a stand-alone introduction to Born-Oppenheimer Molecular Dynamics (BOMD) simulations. The reader is referred to the textbook of Frenkel and Smit [2] for a review of the field.

1 Integrating the equations of motion

The MD functionality implemented in ONETEP is founded on the Born-Oppenheimer approximation which states that the electrons are much lighter than nuclei, the dynamics of electrons is much faster compared to the dynamics of the nuclei. As a consequence, the former can be considered to react instantaneously to the motion of the latter. The forces acting on the nuclei are derived from the ground state electronic configuration by means of the Hellmann-Feynamn theorem. The motion of the nuclei is described by the laws of classical mechanics

$$\frac{\partial H}{\partial \mathbf{r}} = -\dot{\mathbf{p}} \quad \text{and} \quad \frac{\partial H}{\partial \mathbf{p}} = \dot{\mathbf{r}} \tag{1}$$

where H is the Hamiltonian (or the total energy) of the system and $\mathbf{r}$, $\mathbf{p}$ are the nuclei positions and conjugate momenta. At each MD steps, the forces on the particles are computed, and the particles positions and momenta are updated according to Newton's equations of motion. Though this is an excellent approximation for many materials, it is important to keep in mind that classical dynamics does not account for quantum phenomena such as zero point motion, tunneling, or quantum fluctuations which may play a significant role in the dynamics of some systems.

In a BOMD simulation, the classical laws of motion are integrated using a finite difference scheme (that usually preserves the symplectic structure of phase space, e.g. the Velocity-Verlet algorithm [3, 4]). For small enough time steps, the particle trajectory becomes independent of the discretization and the total energy of the system is

conserved. At room temperature and in situation close to equilibrium, a time step Δt of a fraction of a femtosecond is usually adopted.

The Velocity-Verlet algorithm corresponds to the following set of four operations:

$$1: \quad \mathbf{v}_{n+1/2} = \mathbf{v}_n + \frac{\Delta t}{2m} * \mathbf{F}_n \tag{2}$$

$$2: \quad \mathbf{r}_{n+1} = \mathbf{r}_n + \Delta t * \mathbf{v}_{n+1/2} \tag{3}$$

$$3: \quad \text{Compute ionic forces } \mathbf{F}_{n+1} \tag{4}$$

$$4: \quad \mathbf{v}_{n+1} = \mathbf{v}_{n+1/2} + \frac{\Delta t}{2m} * \mathbf{F}_{n+1} \tag{5}$$

where subscripts are used to label the MD time steps. This approach yields a reversible integrator that weights correctly the phase space and conserves the phase space volume.

The velocities in eqs. (2)-(5), are the internal (or peculiar) velocities and not the atomic velocities. Internal velocities are used to properly take into account the internal motion of the system, for which the total linear momentum must vanish. When using open boundary conditions, the use of internal velocities ensures that also the total internal angular momentum vanishes. By setting the total linear (angular) momentum to zero at the beginning of a simulation while employing atomic velocities in eqs. (2)-(5), does not guarantee to keep the linear (angular) momentum conserved. This is due to numerical errors that unavoidably modify the initial values. One of the possible drawback is the well-known "flying ice cube effect". The interested reader is referred to Ref. [14] for a comprehensive description. However, before printing out the trajectory info to the `rootname.md` file, the internal velocities are transformed back to the atomic velocities for visualization and post-processing. In the limit of very long time, the ergodic hypothesis is invoked which allows us to derive ensemble averages from the molecular trajectories.

Basic input parameters

The Molecular Dynamics functionality is activated by setting the input parameter `TASK` to `MOLECULARDYNAMICS`. If a fresh calculation is started, the initial nuclear positions are read from the `POSITIONS_ABS` block while the nuclear velocities are obtained from the `VELOCITIES` block. If the latter is not specified, the velocities are drawn from a maxwell-boltzmann distribution at a (user defined) temperature set in the `THERMOSTAT` block (see Thermostats section). The values of Δt is determined by the parameter `MD_DELTA_T`. The number of integration steps is fixed by `MD_NUM_ITER`.

For example, the following set of input parameters instructs the code to

run a 4 ps long BOMD calculation with $\Delta t = 0.8$ fs.

```
                TASK  :  MOLECULARDYNAMICS
           MD_DELTA_T  :  0.8 fs
         MD_NUM_ITER  :  5000
      MD_PROPERTIES  :  T
         MD_RESTART  :  F
```

The flag `MD_PROPERTIES` instructs the code to enter the properties module at each MD steps. During the calculation a file `rootname.md` is generated that contains a summary of the trajectory, such as temperature, energies, nuclear positions, velocities and forces at each MD steps. Additionally, the latest phase space coordinates are stored in the unformatted file `rootname.md.restart`. The flag `MD_RESTART` enables to restart an MD calculation from the phase space coordinates stored in `rootname.md.restart`. It is important to stress here that `MD_NUM_ITER` is an incremental counter. This means that the when starting a fresh calculation the number of MD steps corresponds to `MD_NUM_ITER`, while for a restart calculation the actual number of MD steps is calculated as the difference between `MD_NUM_ITER` and the total number of MD steps completed up to that point. Therefore, if we want to continue the 4 ps long calculation of the previous example for other 4 ps, we would have to set

```
                TASK  :  MOLECULARDYNAMICS
           MD_DELTA_T  :  0.8 fs
         MD_NUM_ITER  :  10000
      MD_PROPERTIES  :  T
         MD_RESTART  :  T
```

2 Thermostats

The `THERMOSTAT` block must be defined for any MD calculation, even when performing microcanonical runs. For equilibration purposes or to extract thermodynamical averages, it is often desirable to sample the canonical ensemble (constant-NVT) rather than the microcanonical one (constant-NVE). In order to achieve this, there needs to be a mechanism (i.e. a thermostat) by which the system can exchange energy with the rest of the universe. Several thermostats, Andersen, Langevin, Nose-Hoover chains, Berendsen and Bussi, are available in ONETEP.

2.1 Andersen thermostat

One of the simplest constant temperature algorithm has been proposed by Andersen [5]. The system is thermally coupled with a bath of fictitious particles at temperature T. Practically this coupling acts by replacing the momentum of a number of atoms by a new momentum derived from the appropriate Boltzmann distribution. The strength of the coupling can be adjusted by fixing the characteristic time (τ) at which the momentum rescaling occurs and the amplitude (γ) of the rescaling. Eventually, the probability that collision occurs during a time step Δt is given by,

$$q_{col} = 1 - e^{-\Delta t/\tau} \tag{6}$$

and the collision on atom i acts as,

$$p^{new} = \sqrt{(1-\gamma^2)}\; p + \gamma\; p^{boltzmann}, \tag{7}$$

where p^{new} is the momentum rescaled by Andersen thermostat, and $p^{boltzmann}$ is a random variable with appropriate Boltzmann distribution.

2.2 Langevin thermostat

The Langevin thermostat accounts for the motion of the atoms in the presence of a fictitious viscous solvent [6]. As they have to be pushed away, the solvent particles create a friction force damping the momentum of the atoms. Besides random perturbations of the ionic forces arise from the collisions between the atoms and the solvent particles. Langevin dynamic corresponds to the modified equation of motion,

$$\dot{p_\alpha} = F_\alpha - \gamma \frac{p_\alpha}{m_\alpha} + f_\alpha \tag{8}$$

where greek superscripts label the nuclei, F_α are the conservative forces acting on the nuclei, γ is the damping factor associated with the solvent viscosity and f_α are the random forces accounting for the collisions. In order to guarantee NVT statistics, the random forces and the damping factor are chosen so as to fulfill the fluctuation-dissipation theorem. Eventually, the update of the nuclei momenta p_α and forces F_α is given by,

$$p_\alpha^{new} = p_\alpha \; * \; e^{-\gamma \Delta t} \tag{9}$$

$$F_\alpha^{new} = F_\alpha \; * \; \frac{1}{\gamma}(1 - e^{-\gamma \Delta t}) + f_\alpha \tag{10}$$

$$f_\alpha = \sqrt{\frac{m^\alpha k_B T (1 - e^{-2\gamma \Delta t})}{\Delta t^2}} * \xi_\alpha \tag{11}$$

where $\{\xi_\alpha\}$ is a set of mutually uncorrelated random Gaussian variables with a zero mean and unit variance.

2.3 Nosé-Hoover thermostat and Nose-Hoover chains

In the Andersen and Langevin approaches, the constant temperature is achieved by stochastic collisions with fictitious particles. The approach of Nosé is different and allows to perform deterministic MD at constant temperature [7, 8]. To achieve isothermal MD, an additional coordinate associated with an effective mass is introduced in the Lagrangian ruling the dynamics of the nuclei. For a derivation of the equations of motion, the reader is referred to the textbook of Berend and Smith. Provided the center of mass of the system remains fixed, the Nosé-Hoover thermostat leads to a canonical distribution of positions and momenta. To alleviate this restriction on the center of mass, the nuclei are coupled to a Nosé-Hoover thermostat whose fluctuations are determined by another thermostat (i.e. the so called Nosé-Hoover chains). In ONETEP, the effective masse of the thermostats (Q_{th_i}) is chosen, following the prescription of Martyna and Tuckerman [10], as

$$Q_{th_1} = 3N\frac{k_B T}{\omega^2} \tag{12}$$

$$Q_{th_i} = \frac{k_B T}{\omega^2}, \tag{13}$$

where N is the number of nuclei and $\omega = 2\pi/\tau$ is the characteristic frequency of the thermostats. That parameter τ has to be chosen so as to guarantee a good coupling with the atomic system. E.g. when water is used as solvent in the system, a value of 9.4 fs is appropriate as it corresponds to the first asymmetric stretching mode of water molecules.

2.4 Berendsen thermostat

In the Berendsen thermostat, the ionic equation of motions are supplemented by a first order equation for the kinetic energy,

$$dK = \frac{K_t - K}{\tau} dt\,, \tag{14}$$

where K_t stands for the target kinetic energy. The weak coupling of the system with the heat bath is determined by the time constant τ. This thermostat does not generate a canonical ensemble but is vary efficient for thermalization of large systems.

2.5 Canonical velocity scaling

An extension of the Berendsen thermostat allows to recover the canonical distribution of the kinetic energy. In this approach, the instantaneous kinetic energy is propagated using an auxiliary stochastic dynamics. The equation of motion for the kinetic energy is defined as,

$$dK = \frac{K_t - K}{\tau}\, dt \, + 2\sqrt{\frac{K K_t}{3N\tau}}\, dW\,, \tag{15}$$

whereK_t stands for the target kinetic energy and dW is a Wiener noise. For a complete derivation of the equations of motion, the reader is referred to G. Bussi et al. [9]. In the same way as for the Berendsen thermostat, the coupling of the system with the heat bath is determined by the characteristic time τ.

Thermostat definition

The parameters related to constant-NVE or constant-NVT sampling are determined by means of the `THERMOSTAT` block. For a constant-NVE calculation, the thermostat block is needed to specify the initial temperature for the maxwell-boltzmann distribution, from which initial velocities are drawn. For constant-NVT sampling, different thermostats can be associated with different groups of atoms.

```
%block thermostat
start_iter   end_iter   thermo_name   temp   ! First thermostat definition
                    option_1 = value         ! Optional parameter 1
                    option_2 = value         ! Optional parameter 2
start_iter   stop_iter  thermo_name   temp   ! Second thermostat definition
                    option_1 = value         ! Optional parameter 1
                    option_2 = value         ! Optional parameter 2
%endblock thermostat
```

A thermostat definition contains four mandatory parameters and several optional parameters. The mandatory parameters are : the starting and stopping MD steps (these must be set bearing in mind the global counter logic), the type of thermostat (i.e. `none`, `andersen`, `langevin`, `nosehoover`, `berendsen`, or `bussi`,) and the temperature. The line containing the mandatory parameters may be followed by one or more of optional parameter definition (one per line).

Let us set an NVT calculation at 300K with Langevin thermostat for the equilibration (3000 steps) and Nosé-Hoover thermostat for the thermodynamical sampling (10000 steps). The input parameters could look like

```
%block Thermostat
    0     3000    langevin       300.0 K
          damp = 0.2
3001    13000    nosehoover     300.0 K
          nchain = 4
          nsteps = 10
          tau = 100 fs
%endblock Thermostat
```

If both `MD_RESTART` and `MD_RESTART_THERMO` flags are set to true, the thermostat internal parameters are initialized from the values found in the unformatted file named`rootname.thermo.restart` (`rootname.thermo.global.restart` if `MD_GLOBAL_RESTART = .true.`, see section on MD history). This is particularly useful when using Nosé-Hoover thermostat as it avoids any disruption in the trajectories of the thermostat coordinates. A formatted report on the thermostat trajectories is outputted in the file `rootname.thermo`.

Thermostat optional parameters

tgrad (Physical) (default = 0K)
Discrete variation of temperature T per MD step.

group (Integer) (default = 0)
Index of the group of atoms (as defined in `positions_abs`) to wich the thermostat is coupled. If no group of atoms is specified the thermostat is applied to the full system (i.e. group index 0).

tau (Physical) (default = 10*`MD_DELTA_T`)
Characteristic time scale of the thermostat. Depending on the type of thermostat, it may relate either to the average collision frequency (see Eq.6) or the thermostat fluctuation frequency (see Eqs. 12 and 13) or to the coupling with the heat bath (see Eqs. 14 and 15).

mix (real) (default = 1.0)
Collision amplitude of the Andersen thermostat (see Eq. 7).

damp (real) (default = 0.2)
Damping factor in the Langevin equation of motion (see Eq. 8).

nchain (integer) (default = 0)
Number of thermostats in the Nosé Hoover chain.

nsteps (integer) (default = 20)
Number of substep used to integrate the equation of motion of the Nosé-Hoover coordinates.

update (logical) (default = .false.)
Impose to update the effective masses of the Nosé-Hoover coordinates when the temperature is modified.

3 Using MD history

In order to predict sensible trajectories and ensemble averages, BOMD requires to solve the self-consistent field (SCF) equations that determines the ground-state electronic structure at each MD steps. Solving the SCF equations therefore dominates the computational effort. The number of SCF cycles required to reach a given level of self-consistency can be substantially reduced by using a good initial guess for the electronic degrees of freedom. Various schemes have been proposed that enable to make a good use of the MD history in order to build efficient initial guesses.

3.1 Extrapolation of NGWFs and density kernel

In ONETEP [1], the Kohn-Sham SCF equations are formulated in terms of the single-particle density matrix $\rho(\mathbf{x}, \mathbf{x}')$,

$$\rho(\mathbf{x}, \mathbf{x}') = \phi_\alpha(\mathbf{x})\ K^{\alpha\beta}\ \phi^*_\beta(\mathbf{x}')\ , \tag{16}$$

where Einstein's notation for repeated indices has been used. $\{\phi_\alpha(\mathbf{x})\}$ is a set of localised support functions, hereafter named Non-orthogonal Generalized Wannier Functions (NGWFs), and $\mathbf{K}$ is the kernel representing the density operator. At each MD step, the total energy is minimized with respect to both the density kernel and the support functions. Here below, we briefly review various algorithms that allows to initialise the density kernel and NGWFs by extrapolation from previous time steps.

Hereafter, χ_i^{init} and χ_i^{scf} are used to represent respectively the initial guess and SCF solution for either the density kernel or a given NGWF at time $t = i\Delta t$.

One-dimensional linear extrapolation :

The simplest attempt at a trial configuration for the electronic degrees of freedom is the linear extrapolation,

$$\chi^{\text{init}}_{(i+1)} = 2\chi^{\text{scf}}_i - \chi^{\text{scf}}_{(i-1)}. \tag{17}$$

Multi-dimensional linear extrapolation :

The idea of multi-dimensional linear extrapolation was first proposed by Arias, Payne and Joannopoulos for the generation of trial wavefunctions [11]. The one-dimensional linear extrapolation scheme creates an acceptable initial configuration for the ionic coordinates $\mathbf{r}' = 2\mathbf{r}_i - \mathbf{r}_{(\mathbf{i-1})}$. However, the actual coordinates $\mathbf{r}_{i+1}$ are in general different. In order to account for the non-linear propagation of the coordinates, the extrapolation can be generalized as follow,

$$\chi^{\text{init}}_{(i+1)} = \chi^{\text{scf}}_i + \sum_{n=0}^{N} c_n \left(\chi^{\text{scf}}_{(i-n)} - \chi^{\text{scf}}_{(i-(n+1))}\right) \tag{18}$$

where the $N+1$ coefficients $\{c_n\}$ are chosen by minimizing the norm,

$$\left\| (\mathbf{r}_i - \mathbf{r}_{i+1}) + \sum_{n=0}^{N} c_n \left(\mathbf{r}_{(i-n)} - \mathbf{r}_{(i-(n+1))} \right) \right\|. \tag{19}$$

This insures that the extrapolated degrees of freedom are in close correspondence to the BOMD trajectory.

Generalized multi-dimensional linear extrapolation :

The multi-dimensional extrapolation of NGWFs can be further generalized in order to account for the local characteristics of the ionic trajectories. By introducing a localization function $F(r - r_{cut})$ within Eq.19, the coefficients $\{c_n\}$ can be further optimized with respect to the local environment. In practice, a set of coefficients $\{c_n\}_\alpha$ is derived for each ion (α) by minimizing the modified norm,

$$\left\| \left(\mathbf{r}'(\alpha)_i - \mathbf{r}'(\alpha)_{(i+1)} \right) + \sum_{n=0}^{m} c(\alpha)_n \left(\mathbf{r}'(\alpha)_{(i-n)} - \mathbf{r}'(\alpha)_{(i-(n+1))} \right) \right\|, \tag{20}$$

where $\mathbf{r}'(\alpha)_i$ refers to a local projection of the ionic coordinates at time t_i,

$$r'(\alpha)_{\beta,i} = F(r_{\alpha,i} - r_{\beta,i} - r_{cut})\ r_{\beta,i} \tag{21}$$

This way, the extrapolated NGWFs associated with a given ion are in better correspondence to the BOMD trajectory of its local environment.

One-dimensional polynomial extrapolation :

Another way to extrapolate the density kernel and NGWFs is to assume that each element of the density kernel $(K^{\alpha\beta})$ and component of the NGWFs on the grid $(\phi_\alpha(\mathbf{x}))$ can be represented as a polynomial in the time t. Applied to the density kernel, this gives,

$$K^{\alpha\beta}(t) = \sum_{m=0}^{N} c_m^{\alpha\beta}\ t^m \tag{22}$$

where the $N+1$ extrapolation coefficients $c_m^{\alpha\beta}$ are dertemined by fitting the polynomial expression to the last $N+1$ values of $K^{\alpha\beta}$.

3.2 Density kernel transformations

The extrapolation schemes, as described above, illustrates a point of view in which the density kernel and the support functions are considered on the same footing, either as a functional of the ionic coordinates or as an oscillatory function in time. This is ignoring the close link between the support functions and the density kernel (see Eq.3.2). There is a broader point of view, where the density kernel $(K^{\alpha\beta})$ is considered as the representation of the density operator in the time-dependent basis formed by the NGWFs. If one assume that the BOMD propagation of the electronic degrees of freedom is more or less adiabatic, it is tempting to rely on the schemes described above for the

extrapolation of the support functions and to transform the latest density kernel in order to account for the modification of the basis set. In ONETEP, this can be done in two ways.

Projection of the density kernel :

The simplest attempt at transforming the density kernel in order to adapt it to the new support functions is to project the density kernel onto the extrapolated NGWFs. This transformation reads,

$$\mathbf{K}^{\text{init}}_{(i+1)} = \left(\mathbf{S}^{\text{init}}_{i+1}\right)^{-1} \mathbf{T}_{(i+1),i}\ \mathbf{K}^{\text{scf}}_{i}\ \mathbf{T}_{i,(i+1)}\left(\mathbf{S}^{\text{init}}_{i+1}\right)^{-1}\ , \tag{23}$$

where $\mathbf{K}_i$ and $\mathbf{S}_i$ stand for the density kernel and overlap matrix at MD step i; and $\mathbf{T}_{i,(i+1)}$ is the overlap between the NGWFs at MD step i and the *extrapolated* NGWFs at step $(i+1)$.

Christoffel correction to the density kernel :

While projecting the density kernel onto the extrapolated support functions is appealing because of its conceptual simplicity, it does not fully account for the tensorial character of the density operator. As the support functions are extrapolated, the metric of the representation manifold changes giving rise to non-vanishing Christoffel symbols. In order to preserve tensorial integrity and idempotency to first order, contributions from the Christoffel symbols should be accounted for in the transformation of the density kernel. The correction to the density kernel then reads,

$$\Delta\mathbf{K}^{\text{init}}_{(i+1)} = -\left(\mathbf{S}^{\text{scf}}_{i}\right)^{-1}\ \mathbf{D}_{(i+1),i}\ \mathbf{K}_i\ -\mathbf{K}_i\ \mathbf{D}_{i,(i+1)}\ \left(\mathbf{S}^{\text{scf}}_{i}\right)^{-1} \tag{24}$$

with

$$\left(\mathbf{D}_{(i+1),i}\right)_{\alpha\beta} = \left\langle (\phi^{\text{init}}_{(i+1)})_\alpha - (\phi^{\text{scf}}_{i})_\alpha \,\middle|\, (\phi^{\text{scf}}_{i})_\beta \right\rangle\ . \tag{25}$$

3.3 Extended Lagrangian propagation of density kernel schemes

Extended Lagrangian naïve approach :

The number of SCF iterations needed to reach a given threshold at each step of the BOMD calculation can be significantly reduced by the extrapolation schemes presented in sections 3.1 and 3.2. However, those methods come with a caveat that has to be kept in mind. While, with a perfect SCF optimization, the SCF ground-state electronic structure is independent from the initial guess, in practice, self-consistence is only achieve up to a given threshold. The consequence of this *incomplete* convergence is that the extrapolation schemes introduce a *memory* effect in the simulation and break the time-reversibility of the BOMD algorithm. As a consequence, the resulting trajectories suffers from systematic error and a significant energy drift may appear on time scales of a few picoseconds. A simple way to restore energy conservation is to impose tighter SCF convergence thresholds. However, this may results in a considerable increase of the computational cost. Another solution has been proposed by Niklasson et al. (see Ref.[12]). This scheme

restores the time-reversibility of BOMD by extending the BO Lagrangian with auxilliary degrees of freedom directly associated with χ^0, the initial guess of the electronic degrees of freedom. The user is referred to Refs.[12, 13] for a complete introduction to this formalism.

Extended Lagrangian with dissipation, dEL/SCF :

A more stable propagation scheme for the density kernel has also been proposed by Niklasson [13]. In this scheme, the numerical errors arising from an incomplete convergence are averaged out via a dissipative term in the extended BO Lagrangian. Following Bowler[15], we propagate the orthogonal representation $\mathbf{P}$ of the auxiliary density kernel, i.e. $\mathbf{P}$ has the sparsity pattern of $\mathbf{KS}$ rather than of $\mathbf{K}$, to avoid the extra intricacies of propagating tensors in a space with non-unitary metric. The dissipative term is defined in terms of a linear combination of previous density kernels, which using the symplectic Verlet algorithm, yields the following equation of motion

$$\begin{aligned} \mathbf{P}_{i+1} &= 2\mathbf{P}_i - \mathbf{P}_{i-1} + \kappa[(\mathbf{KS}^{\text{scf}})_i - \mathbf{P}_i] \\ &\quad + \alpha \sum_{m=0}^{M} c_m \mathbf{P}_{i-m}. \end{aligned} \tag{26}$$

where κ, α and c_m's are optimized coefficients obtained from Ref. [13]. The initial guess for the density kernel is given by

$$\mathbf{K}_{i+1}^{\text{init}} = \text{sym}(\mathbf{PS}_{i+1}^{-1}) = \frac{1}{2}[(\mathbf{PS}^{-1})_{i+1} + (\mathbf{S}^{-1}\mathbf{P})_{i+1}] \tag{27}$$

The problem with the above mentioned scheme lays in the use of a dissipative term that unavoidably breaks the time-reversibility, which in turn will generate, over long simulation time, a drift in the energy. However, for simulation time accessible at the moment in AIMD, this issue is of little concern.

Extended Lagrangian with thermostat, inertial iEL/SCF :

Recently, a similar scheme that overcomes the issue of the time breaking symmetry has been proposed [16]. The idea is to control the dynamics of the auxiliary degrees of freedom through a thermostat. One of the simplest yet efficient thermostats around is the Berendsen thermostat. Here, we also propagate the orthogonal representation of the auxiliary density kernel for the same reasons listed in the previous section. The equation of motion for the auxiliary density kernel, using a velocity-Verlet integrator, reads

$$\mathbf{P}_{i+1} = \mathbf{P}_i + \dot{\mathbf{P}}_i \Delta t + \omega^2 \Delta t^2 [(\mathbf{KS})_i^{\text{scf}} - \mathbf{P}_i] \tag{28}$$

$$\begin{aligned} \dot{\mathbf{P}}_{i+1} &= \gamma_i \tilde{\dot{\mathbf{P}}}_{i+1} \\ &= \gamma_i \{\dot{\mathbf{P}}_i + \omega^2 \Delta t/2 [((\mathbf{KS})_{i+1}^{\text{scf}} - \mathbf{P}_{i+1}) + \\ &\quad ((\mathbf{KS})_i^{\text{scf}} - \mathbf{P}_i)]\} \end{aligned} \tag{29}$$

with γ_i given by

$$\gamma_i = \sqrt{1 + \frac{\tau}{\Delta t}\left(\frac{T_{\rm K}}{\left\langle \dot{\mathbf{P}}_i^{\,2} \right\rangle} - 1\right)} \tag{30}$$

where $T_{\rm K}$ is the target temperature, τ is the characteristic time of the thermostat, and $\left\langle \dot{\mathbf{P}}_i^{\,2} \right\rangle$ is the instantaneous temperature of the auxiliary degrees of freedom. The key parameter is the target temperature and much care must be done in assigning a value for it.

Extrapolation and propagation of NGWFs

The main input parameters that determine the extrapolation and propagation of NGWFs are `mix_ngwfs_type` and `mix_ngwfs_num`. The localization function $F(r - r_{cut})$ used in the generalized version of the multi-dimensional linear extrapolation (see Eq. 21.) is characterized by the input parameters `mix_local_length` and `mix_local_smear`.

mix_ngwfs_type (String) (default = none)

none	:	No use of MD history. Initial NGWFs are built according to `species_atomic_set` block.
reuse	:	No mixing of NGWFs. NGWFs at previous MD step are used as initial guess.
linear	:	One dimensional linear extrapolation from NGWFs at two previous MD steps (see Eq.17).
multid	:	Multi-dimensional linear extrapolation from NGWFs at previous MD steps (see Eqs. 18 and 19). The dimension of the extrapolation space is determined by input parameter `mix_ngwfs_num`.
poly	:	One-dimensional polynomial extrapolation from NGWFs at previous steps (see Eqs. 20). The degree of the extrapolation polynom is determined by input parameter `mix_ngwfs_num`.
local	:	Generalized multi-dimensional linear extrapolation from NGWFs at previous steps (see Eqs. 20). The dimension of the extrapolation space is determined by input parameter `mix_ngwfs_num`. The localization radius is determine by input parameter `mix_local_length`. Optionnally, the localization radius can be smeared out by using non-zero values for `mix_local_smear`

trprop : Time-reversible propagation of auxiliary NGWFs. See section 3.3 and references therin.

mix_ngwfs_num (Integer) (default depends on **mix_ngwfs_type**)
Number of previous MD steps required to build the initial guess for the density kernel.

mix_loc_length (Physical) (default = 10.0 bohr)
Cutoff radius of the localization function $F(r - r_{cut})$ see Eq. 21.

mix_loc_smear (Physical) (default = 5.0 bohr)
When `mix_loc_smear` is non-vanishing, the localization function $F(r-r_{cut})$ is assumed to be Fermi-Dirac like with a characteristic smearing of `mix_loc_smear`.

Extrapolation and transformation of density kernel

The main input parameters that determine the extrapolation, transformation and propagation schemes for the density kernel and NGWFs are respectively `mix_dkn_type` and `mix_dkn_num`.

mix_dkn_type (String) (default = none)

none : No use of MD history. Initial density kernel is built according to `coreham_denskern_guess` parameter.

reuse : No kernel mixing. SCF density kernel at previous MD step is used as initial guess.

linear : One dimensional linear extrapolation from density kernel at two previous MD steps (see Eq.17).

multid : Multi-dimensional linear extrapolation from density kernel at previous MD steps (see Eqs. 18 and 19). The dimension of the extrapolation space is determined by `mix_dkn_num`.

poly : One-dimensional polynomial extrapolation from density kernel at previous steps (see Eqs. 20). The degree of the extrapolation polynom is determined by `mix_dkn_num`.

proj : Projection of the previous SCF density kernel onto the set of extrapolated NGWFs. This option requires `mix_ngwfs_type` $\neq$ none.

tensor : Correction of the previous SCF density kernel in order to preserve tensorial integrity. This option requires `mix_ngwfs_type` $\neq$ none.

trprop : Naïve time-reversible propagation of auxiliary density kernel. See section 3.3 and references therin.

dissip : Dissipative propagation of auxiliary density kernel. See section 3.3 and references therin. The number of previous MD steps used for the derivation of the dissipative force is determined by `mix_dkn_num`.

berendsen : Thermostatted propagation of auxiliary density kernel with Berendsen thermostat. See section 3.3 and references therein. The target temperature for the thermostat is set by `md_aux_dkn_t` and the characteristic time constant τ by `md_aux_beren_tc`.

mix_dkn_num (Integer) (default depends on **mix_dkn_type**)
Number of previous MD steps required to build the initial guess for the density kernel.

mix_aux_dkn_t (Physical) (default = 1e-8)
Target temperature of the auxiliary degrees of freedom to use in the the berendsen propagation of the density kernel.

mix_aux_beren_tc (Physical) (default = 0.1 ps)
Characteristic time constant for the Berendsen thermostat to use in the berendsen propagation of the density kernel.

Additional notes on extrapolation and propagation

Most of the extrapolation and propagation schemes suffer from restricted stability under incomplete SCF convergence. Depending on the convergence parameters, significant discrepancies between the MD trajectories and the Born-Oppenheimer surface may arise during the first few MD iterations. In this case, it is recommended not to use the extrapolation and propagation schemes until a good level of SCF convergence is reached. The

input parameters `mix_ngwfs_init_type` and `mix_ngwfs_init_num` allows to set up a smooth initialization phase. It is also possible to have a different (usually tighter) thresholds during this initialization phase. In fact, the usual `lnv_threshold_orig` and `ngwf_threshold_orig` are used to set the LNV and NGWFs gradient thresholds during the intialization phase, while the two keywords `md_lnv_threshold` and `md_ngwf_threshold` determine the LNV threshold and NGWFs gradient threshold for the remaining MD calculation.

It is also possible to periodically reset the MD history using `mix_ngwfs_reset`, `mix_dkn_reset` and `md_reset_history`, although this is not recommended if one wants to avoid jumps into the energy profile, i.e. avoid discontinuities in energy plots.

md_reset_history (Integer) (default = 100)
Every n MD steps, new initial guesses for the electronic degrees of freedom are built according to `coreham_denskern_guess` and `species_atomic_set`.

mix_ngwfs_reset (Integer) (default = 50)
Every n MD steps, the NGWFs mixing/extrapolation scheme is reset and a new initial guess for the NGWFs is built according to `species_atomic_set`.

mix_dkn_reset (Integer) (default = 50)
Every n MD steps, the density kernel mixing/extrapolation scheme is reset and a new initial guess for the kernel is built according to `coreham_denskern_guess`.

mix_ngwfs_init_num (Integer) (default = 0)
Length of the initialization phase. Number of MD steps before the activation of the extrapolation/propagation scheme for building NGWFs initial guesses.

mix_ngwfs_init_type (String) (default = none)
- none : During the initialization phase, initial NGWFs are built according to `species_atomic_set` block.
- reuse : During the initialization phase, NGWFs at last MD step is used as initial guess.

mix_dkn_init_num (Integer) (default = 0)
Length of the initialization phase. Number of MD steps before the activation of the extrapolation/propagation scheme for building density kernel initial guesses.

mix_dkn_init_type (String) (default = none)
- none : During the initialization phase, initial density kernels are built according to `coreham_denskern_guess`.
- reuse : During the initialization phase, density kernel at last MD step is used as initial guess.

md_lnv_threshold (Double) (default = `lnv_threshold_orig`)
LNV threshold for the MD calculation. This can be set to be different from the initial LNV threshold `lnv_threshold_orig` of the first n steps (set by `mix_ngwfs_init_num`/`mix_dkn_init_num`).

md_ngwf_threshold (Double) (default = `ngwf_threshold_orig`)
NGWFs gradient threshold for the MD calculation. This can be set to be different from the initial NGWFs gradient threshold `ngwf_threshold_orig` of the first n steps (set by `mix_ngwfs_init_num`/`mix_dkn_init_num`).

Additional notes on restart when using a propagation scheme

If a "history" of NGWFs/density kernels is generated during a MD calculation it can be periodically saved into external files through the keyword `md_write_history`. More precisely, when using `md_write_history = T` all the information about the dynamical state (positions, velocities and accelerations), the thermostat state, and the propagation scheme is saved to external files as well. To restart a MD calculation by reading in the history from the last save the `md_global_restart` keyword must be set to true in the restart input file. On a restart, one can either use the thermostat state stored in `rootname.thermo.global.restart` or start with a new thermostat block. This is achieved by setting the `md_restart_thermo` keyword.

md_write_history (Integer) (default = -1)
Every n MD steps the history of auxiliary density kernels is written into external files `rootname.history.dkn#.scf/init/vel` (one of each kind for any element in the history). The info on the dynamical state, the thermostat, the propagation scheme and the composition method are saved into `rootname.md.global.restart`, `rootname.md.thermo.restart`, `rootname.history.info` and `rootname.history.var` respectively.

md_global_restart (Logical) (default = false)
MD global restart. This allows to restart a calculation by reading in a history of density kernels if present. `md_restart` is set to false.

md_restart_thermo (Logical) (default = true)
Read thermostat info from file. If set to false, the thermostat is set according to the thermostat block in the input file.

WARNING: Restarting a calculation with `md_global_restart = T` comes with a caveat: depending on the value of `md_write_history` the last batch of NGWFs/density kernels saved to files may not correspond to the NGWFs/density kernels history of the last MD step completed. However, the calculation restarts using the information stored in the `rootname.history.info` and `rootname.history.var`. As a result, there might be duplicated entries in the `rootname.md` file which has to be deleted manually by the user.

For example, let's consider the following scenario

```
MD_NUM_ITER  :  124
MD_WRITE_HISTORY  :  10
```

where we save a history of density kernels every 10 MD steps and the simulation stops after 124 steps. The last batch of density kernels (together with all the other MD info) is saved at step 120, but the summary of the trajectory from step 121-124 is still appended to `rootname.md`. When restarting with `md_global_restart = T`, the code reads in the files `rootname.history.info` and `rootname.history.var` containing the info corresponding to step 120 and starts to append the trajectory info to `rootname.md`. As a result, the summary of the trajectory from step 121-124 in the `rootname.md` is duplicated.

References

[1] C.-K. Skylaris et al., J. Chem. Phys. **122**, 084119 (2005) .

[2] *Understanding Molecular Simulation*, 2nd Ed. D. Frenkel and B. Smit, Academic Press (2001)

[3] L. Verlet, Phys. Rev. **159**, 98 (1967).

[4] W. C. Swope et al., J. Chem. Phys. **76**, 637 (1982).

[5] H. C. Andersen, J. Chem. Phys. **72**, 2384 (1980).

[6] G. S. Grest and K. Kremer, Phys. Rev. A, **33** 3628 (1986).

[7] S. Nose, J. Chem. Phys., **81** 511 (1984).

[8] W. G. Hoover, Phys. Rev. A, **31** 1695 (1985).

[9] G. Bussy et al., J. Chem. Phys., **126** 014101 (2007).

[10] G.J. Martyna, M.E. Tuckerman, et al., Molecular Physics **87**, 1117 (1996)

[11] T. A. Arias et al., Phys. Rev. B, **45**, 1538 (1992).

[12] A. M. N. Niklasson et al., Phys. Rev. Lett. **97**, 123001 (2006).

[13] A. M. N. Niklasson et al., J. Chem. Phys. **130**, 214109 (2009).

[14] P. H. Hünenberger, Advanced Computer Simulation **173**, 105-109 (2005)

[15] M Arita et al., J. Chem. Theory Comput., **10**, 5419-5425 (2014)

[16] A. Albaugh et al, J. Chem. Phys., **143**, 174104 (2015)