QUANTUM ESPRESSO FOUNDATION

MAX School on Advanced Materials and Molecular Modelling
with QUANTUM ESPRESSO

QE-2021: Hands-on session – Day-8

(Ab Initio Molecular Dynamics)

Riccardo BERTOSSA, Davide Tisi, Paolo Pegolo,
Praveen Chandramathy Surendran, Tao Jiang

MAX
The Abdus Salam International Centre for Theoretical Physics
CINECA
上海大学 SHANGHAI UNIVERSITY
Jožef Stefan Institute Ljubljana, Slovenia
arnes
cecam
SISSA

QE-2021: Hands-on session – Day-8

In this tutorial we will see how to use the basic features of the code `cp.x`. The main features of the code are

1. ground state with direct minimization of the total DFT energy
2. Born-Oppenheimer molecular dynamics (BOMD)
3. Car-Parrinello molecular dynamics (CPMD)
4. variable cell calculations with a barostat
5. Nose Hover thermostat
6. Γ point only
7. ...

QE-2021: Hands-on session – Day-8

Our main goal is to calculate the properties of a system at a given temperature and volume. In this tutorial we are going to:

1. start a BOMD simulation with conjugate gradient minimization of the DFT energy
2. run a Car-Parrinello molecular dynamics simulation (CPMD) with the initial ground state taken from the previous BOMD
3. run the Nose-Hoover thermostat to equilibrate the system at a given temperature
4. run a longer microcanonical ensemble simulation to calculate the properties of the system
5. view the trajectory and calculate $g(r)$ and mean square displacement from the trajectory

Our system will be 8 water molecules.

BOMD: small recap

- trajectory: time series of position, velocities and in general many quantities that depend on time. It is written on many files by the code
- Born Oppenheimer approximation: nuclei are classical. We always use $F = ma$ for them
- adiabaticity: electrons are always in their ground state at every timestep

Start of the simulation

BOMD: we are going to integrate the following equation of motions:

$$M_I \ddot{R}_I = -\nabla_{R_I} E_{DFT}(\{R_J\}_{J \in \text{nuclei}}) \tag{1}$$

That will be discretized according to a Verlet scheme. We have to select all the parameters that are needed to evaluate the forces via the Hellman-Feynman theorem. That means selecting plane wave cutoff, pseudopotentials, etc. No K points are implemented

BOMD cp.x’s input file

```
&control
title = ’Water 8 molecules’,
calculation = ’cp’,
restart_mode = ’from_scratch’,
ndw = 50,
nstep = 100,
iprint = 10,
isave = 1000,
tprnfor = .TRUE.,
dt = 5.0d0,
prefix = ’h2o’,
pseudo_dir=’../pseudo’,
outdir=’./’,
/
```

```
&system
ibrav=1, celldm(1)=13.00000
nat = 24,
ntyp = 2,
ecutwfc = 50.0,
/
&electrons
electron_dynamics = ’cg’,
/
&ions
ion_dynamics = ’verlet’,
ion_velocities = ’random’,
tempw=600.d0
/
```

check for convergence of `ecufwfc`. Choosing a small `dt` at the very beginning can be useful, then increase it.

BOMD cp.x’s input file

```
&cell
cell_dynamics = ’none’, /
ATOMIC_SPECIES
O 16.0d0 O_ONCV_PBE-1.2.upf
H 1.0079d0 H_ONCV_PBE-1.2.upf
ATOMIC_POSITIONS (bohr)
O 0.48E+01 0.37E+01 0.37E+01
H 0.40E+01 0.59E+01 0.35E+01
.
.
.
```

put here all the atomic positions

then, at the end of the input file

```
AUTOPILOT
on_step=10 :  dt = 20.d0
on_step=90 :  dt = 5.d0
ENDRULES
```

This will change the dt parameter during the simulation. I use for the last steps the same `dt` that I'm going to use later for the CP. If the `dt` of the next simulation is different you have to tell the code with `change_step`!

The documentation for the AUTOPILOT module can be found at `https://gitlab.com/QEF/q-e/-/blob/develop/CPV/Doc/README.AUTOPILOT` It is possible to modify some parameters on the fly while the simulation is running by placing a special file called `pilot.mb` inside the folder where you are running the simulation

cp.x's input file

From the `cp.x` input description:

```
Variable:       dt

Type:           REAL
Default:        1.D0
Description:    time step for molecular dynamics, in Hartree atomic
                units
                (1 a.u.=2.4189 * 10^-17 s : beware, PW code use
                 Rydberg atomic units, twice that much!!!)
```

cp.x's input file

From the `cp.x` input description (namelist electrons):

```
Variable:     electron_velocities

Type:         CHARACTER
Description: 'zero'        : restart setting electronic velocities to
                              zero
             'default'     : restart using electronic velocities of the
                            previous run
             'change_step' : restart simulation using electronic
                            velocities of the previous run, with
                            rescaling due to the timestep change.
                            specify the old step via "tolp" as in
                            tolp = 'old_time_step_value' in au.
                            Note that you may want to specify
                            "ion_velocities" = 'change_step'
```

BOMD run and tips on initial state

Ok, now try to run the input file on the cluster!

```
remote_mpirun cp.x -in cp.water8.1-bomd.in
```

See what was produced

- download results from the cluster

- you see a number of files (everything in Hartree atomic unit):
  - `h2o.cel` file that contains the transposed cell vectors
  - `h2o.pos` unwrapped positions of the atoms, same atomic order of the input files (!)
  - `h2o.vel` atomic velocities, same atomic order of the input file (!)
  - `h2o.evp` thermodynamic data. At first you should look at this
  - ...

h2o.evp

First line of file `# nfi time(ps) ekinc T_cell(K) Tion(K) etot enthal econs econt Volume Pressure(GPa)`

- `ekinc` $K_{ELECTRONS}$
- `enthal` $E_{DFT} + PV$
- `etot` E_{DFT} potential energy of the system
- `econs` $E_{DFT} + K_{NUCLEI}$ this is something that is a constant of motion in the limit where the electronic fictitious mass is zero. It has a physical meaning.
- `econt` $E_{DFT} + K_{IONS} + K_{ELECTRONS}$ this is a constant of motion of the lagrangian. If the dt is small enough this will be up to a very good precision a constant. It is not a physical quantity, since $K_{ELECTRONS}$ has *nothing* to do with the quantum kinetic energy of the electrons.

Note that if the verlet algorithm is not used there is no $K_{ELECTRONS}$, since they don't have a velocity defined

See the trajectory

To see the trajectory we convert the output file to something that is readable from, for example, ovito. In the VM there is a simple python script that you can call for this purpose:

```
./analyze.sh
```

then you can open the file `h2o.lammpstrj` with ovito:

```
ovito h2o.lammpstrj
```

Now you can see the atoms! They didn't move a lot with such a small number of steps...

restart files

The `ndw` (number directory write) specify a part of the name of the folder where restart files are written. Note that the wavefunctions of the last two steps are saved, to allow the next verlet algorithm to restart with the correct wfc velocity.

NB: since the code does not read the old timestep from the restart files, if you change the timestep in the newer run you have to tell the code what was the old timestep with the options showed before.

Wavefunction's velocity

To start to integrate the CP equation of motions we need to set the velocities of the nuclei (and those are simple) and the velocity of the electronic degrees of freedom. We cannot directly use the 2 wavefunctions at different timesteps because the DFT solver can choose any phase that it want in the two points (remember that there are unitary transformations that do not change the expectation values of the observables). So we fix this freedom by using the *parallel transport gauge*

$$P(t) = \sum_v |\psi_v(t)\rangle\langle\psi_v(t)| \tag{2}$$

$$|\psi\rangle = P|\psi\rangle \tag{3}$$

$$|\dot{\psi}\rangle = \dot{P}|\psi\rangle + P|\dot{\psi}\rangle \tag{4}$$

$$\tag{5}$$

The parallel transport gauge is equivalent to say that $P|\dot{\psi}\rangle = 0$.

Wavefunction's velocity

It is possible to compute in a consistent way the wavefunctions at two consecutive timesteps in BOMD, and use those ground state to initialize the verlet algorithm for the CP equation of motion.

$$P(t) = \sum_v |\psi_v(t)\rangle\langle\psi_v(t)| \tag{6}$$

$$|\psi(t)\rangle = |\psi(t-dt)\rangle + dt\dot{P}|\psi(t-dt)\rangle \tag{7}$$

The numerical time derivative of the projector $P(t)$ is well defined and does not depend on the gauge of $|\psi\rangle$.

Note: now (2021-05-26) this is implemented only when using CG with norm conserving pseudo

CP equation of motions

The CP lagrangian is:

$$\sum_v \frac{1}{2}\mu \int_\Omega d^3r|\dot{\psi}_v|^2 + \sum_I \frac{1}{2} M_I \dot{R}_I^2 - E_{DFT}[\{\psi_v\},\{R_I\}] + \sum_{i,j} \Lambda_{ij} \left(\int_\Omega d^3r \psi_i^* \psi_j - \delta_{ij} \right) \tag{8}$$

From that you can derive the equation of motions that are:

$$\mu \ddot{\psi}_v(r,t) = -\frac{\delta E}{\delta \psi_v^*}(r,t) + \sum_j \Lambda_{vj}\psi_j \tag{9}$$

$$M_I \ddot{R}_I = -\nabla_{R_I} E \tag{10}$$

μ is a parameter, Λ_{ij} are the lagrangian multipliers needed to keep the wavefunctions orthonormal. In the limit where $\mu \to 0$ the approximation is exact, but you'll have to use an infinitely small timestep.

CPMD start

Now we will modify the input file of the BOMD to do the verlet integration of the Car-Parrinello lagrangian. Few modifications are necessary:

- set `calculation = 'restart'` to start from a previously stopped calculation
- set `ndr = 50` (set the folder that the code will use to read the restart file)
- set `ndw = 51` (increase by one the number of the folder where the code will write the restart file)
- set `nstep = 1000` if you want to run for 1000 steps
- set `electron_dynamics = 'verlet'` to set the verlet algorithm to integrate the Car-Parrinello equation of motion
- set `emass = 50.d` μ parameter
- set `ion_velocities = 'default'` to read the velocity from the specified restart file

- remove the `AUTOPILOT` card

Output produced

After syncing again the files, look at `h2o.evp`. Now you see that the `ekinc` column is not zero! Verify that the constant of motion is conserved with a good approximation. That means that the highest frequency of the system are sampled with a reasonable rate during the integration of the equation. Remember that we have fast oscillating electronic degrees of freedom. Verify what happens to the instantaneous temperature of the system.

```
$ gnuplot
gnuplot> load 'plot_thermo.gp'
```

If you look again at the trajectory (after executing again the commands to convert it), you'll see that it's longer. Every time the trajectory data is appended to the output files `h2o.???`

Maybe you noticed the file `h2o.for`. This contains the computed force, and is printed only if in the input you have `tprnfor = .true.`. Note that there is a factor two between `cp.x`'s forces and `pw.x`'s one. Do you think that they are more or less the same?

Nose-Hoover thermostat

To have the system equilibrated at a particular temperature we use a thermostat. We have to do the following changes to the input file:

- set the number of steps to 5000 with `nstep = 5000`
- increase by one `ndr` and `ndw`
- set the Nose-Hoover thermostat (namelist `IONS`):
  - set nose: `ion_temperature = 'nose'`
  - temperature: `tempw = 600.d0`

Change the input file and submit it to the cluster!

See what happened

Now it is possible to follow again the same steps to inspect the result of the simulation. Always keep an eye to the constant of motion and to the electronic fictitious kinetic energy!

optional CG step

Before proceding with the "production" NVE simulation, usually it is a good idea to minimize again the electronic ground state and recompute again the wavefunctions velocities with the projection trick. I stress that this step is not necessary, but it can be used in more problematic simulations to keep the electrons cold. So let's modify again the input file. As usual increase by one `ndr` and `ndw`, then:

- set the number of steps to 1 with `nstep = 1`
- increase by one both `ndr` and `ndw`
- change `electron_dynamics = 'cg'`
- change `ion_temperature = 'not_controlled'`

In this way we start again with the electronic wavefunctions "from scratch". Run it on the cluster, it should be very fast.

production NVE simulation

Now we start the final NVE run, that will be the longest. No news here, it is the same as the previous verlet run, but with different `ndr`, `ndw` and `nstep`

- increase by one both `ndr` and `ndw`
- set `nstep = 10000`
- set again verlet for electrons `electron_dynamics = 'verlet'`

submit this calculation and wait for the result.

look at the trajectory

Wow, after copying the data back to our local directory we find out a lot of trajectory! You can convert it to the LAMMPS format using the known python script `cp2lammps.py` and look at it. Woah, the atoms are going outside of the simulation cell!

Lets have a look at the results. I prepared for you a script that does everything:

```
./analyze.sh
```

$g(r)$ plot

Now it is time to calculate something from the trajectory that the cluster computed for us. We will use the C++ code `https://github.com/rikigigi/analisi` that is already installed inside the VM. This code reads the LAMMPS binary format (the binary format is by far faster to load). We generated with the `analyze.sh` script the pair distribution function $g(r)$ plot inside the file `gofr`. It is defined as

$$g_{ab}(r) = \frac{1}{N_a N_b} \sum_{i=1}^{N_a} \sum_{j=1}^{N_b} \langle \delta(|\mathbf{r}_{ij}| - r) \rangle \tag{11}$$

To visualize it you can type the following commands inside gnuplot:

```
$ gnuplot
gnuplot> load 'plot_gofr.gp'
```

mean square displacement plot

The mean square displacement is defined as

$$msd(t) = \langle (R_i(t) - R_i(0))^2 \rangle \tag{12}$$

where is possible to do an average over all atoms of the same type. You can see the MSD plot. This is useful for knowing if the system is diffusive (so it may be a liquid) or not.

Visualize it with gnuplot:

```
$ gnuplot
gnuplot> load 'plot_msd.gp'
```

force comparison between CP and BO

(OPTIONAL) You can, as an experiment, try to pick a timestep, copy and paste atomic positions inside a `pw.x` input file and compute the force. Then plot forces CP vs forces PW. How are the ratios distributed?