Introduction path integral molecular dynamics using i-PI

Venkat Kapil, Bingqing Cheng

June 2016

In this set of exercises we will learn about the basics of doing path integral molecular dynamics simulations using i-PI. We will begin by performing conventional molecular dynamics simulation of classical nuclei, and move towards highly quantum mechanical systems. During the process, the design principles, features and input/output format of i-PI will be introduced.

Exercise 1 Molecular Dynamics simulations doing in a different way: server and clients

We will start by giving a short and simple introduction on the basics of molecular dynamics. Under the Born-Oppenheimer approximation, the dynamics of the electronic degrees of freedom is completely decoupled from that of the nuclei, and the electrons always occupy the ground state for each configuration of the nuclei. In a fully classical treatment of nuclei, an additional assumption is made that their dynamics can be described in terms of the Hamiltonian

$$H(\mathbf{p},\mathbf{q}) = \sum_i \frac{\mathbf{p}_i^2}{2m_i} + V(\{\mathbf{q}_i\}), \tag{1}$$

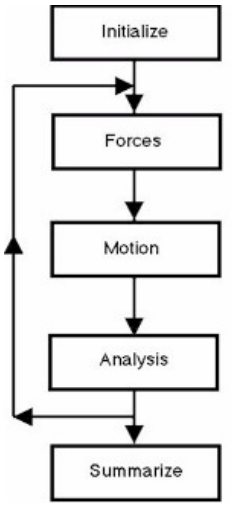
where m_i, $\mathbf{p}_i$ and $\mathbf{q}_i$ are the mass, the momentum and the coordinates of the i-th nucleus, respectively.

In order to simulate the time evolution of particles in a system with this Hamiltonian, we need to take a couple of steps. The equation of motion can be written as

$$\frac{d\mathbf{p}_i}{dt} = -\frac{\partial H(\mathbf{p},\mathbf{q})}{\partial \mathbf{q}_i} = -\frac{dV(\{\mathbf{q}_i\})}{d\mathbf{q}_i} \tag{2}$$

$$\frac{d\mathbf{q}_i}{dt} = \frac{\partial H(\mathbf{p},\mathbf{q})}{\partial \mathbf{p}_i} = \frac{\mathbf{p}_i}{m_i}. \tag{3}$$

The time domain can then be discretized into small and finite time steps with length Δt. The key processes of a molecular dynamics run are illustrated in the flowchart below. At the initialization stage, the program is informed about some basic information about the system, such as the number of atoms, their types, system size, etc. After that, the program goes through the loop "Forces $\rightarrow$ Motion $\rightarrow$ Analysis" once per discrete simulation time step. The force acting on each atom

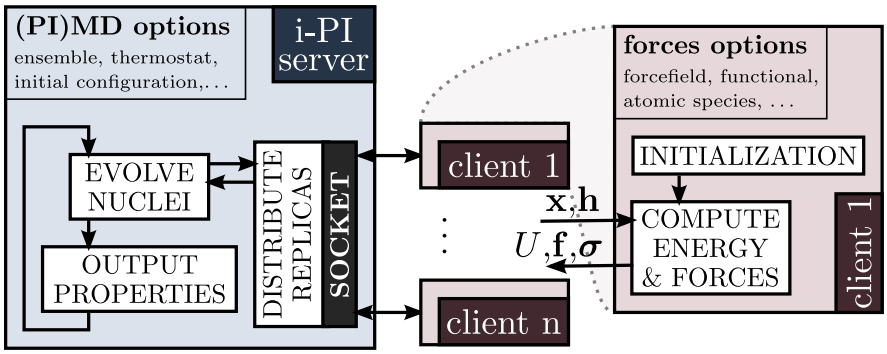


$\mathbf{f}_i = -\dfrac{\partial V}{\partial \mathbf{q}_i}$ is evaluated first, and are then used to advance the momenta and the coordinates of the nucleus.

A simple integrator for the equations of motion (which is not generally used by any program) that evolves the position and momenta in time is given by $\mathbf{p}_i(t + \Delta t) \leftarrow \mathbf{p}_i(t) + \mathbf{f}_i \Delta t$ and $\mathbf{q}_i(t + \Delta t) \leftarrow \mathbf{q}_i(t) + \dfrac{\mathbf{p}_i}{m_i}\Delta t$. In practice most programs use the Verlet integrator or higher order integrators which are much more accurate.

At the analysis step, the properties of interest for the system can be computed, stored, or analyzed on the fly.

From the flowchart above, one can observe that each block of the program can be easily modularized. One can, for example, carry out the force evaluation using one program, and only communicate the results to another program which performs the motion and the analysis parts. This is precisely the philosophy behind i-PI, which acts as a server that takes care of the evolution of nuclei and the evaluation of properties of the system, having as clients other programs that compute forces and potential energies. An graphical representation of how i-PI works is given in the scheme below.



In this exercise, we will perform a simple Molecular Dynamics (MD) calculation. Instead of doing the simulation directly using out-of-the-box MD software such as LAMMPS, we are going to complete the task in a slightly more complex way. We are going to use i-PI in conjunction with LAMMPS to demonstrate the work principles of i-PI as well as the subtle advantages that it brings. The simulation will be performed on one gas phase water molecule using NVT ensemble at 300K. We will also record the kinetic and potential energy of the system.

1. Read the LAMMPS input file `ex-1/300K-G-h2o/in.water`. The force field specifications

   ```
   pair_style lj/cut/tip4p/long 1 2 1 1 0.278072379 17.007
   bond_style class2
   angle_style harmonic
   kspace_style pppm/tip4p 0.0001
   ```

   means that a q-TIP4P force field is being used [**?**].
   On the parts of the input file that specifies how motions are propagated, this line

   ```
   fix 1 all ipi 300K-G.0 32345 unix
   ```

   is used, which basically means LAMMPS will only be in charge of the force evaluation and send that information to i-PI. The latter will take care of the motion and the analysis parts. All the information will be transmitted through a UNIX-domain socket with the name `300K-G.0`, which is a mechanism for local, inter-process communication.

2. Now open the i-PI input file `input.xml` in the same folder. For a moment let us ignore all the other entries but only focus on the followings

   ```
   <ffsocket name="tip4p" pbc="true" mode="unix">
   .. .. <address> 300K-G.0 </address>
   </ffsocket>
   ```

   This indicates that i-PI will receive any information sent by LAMMPS through the UNIX-domain socket with the tag `300K-G.0`, and send the coordinates of the nuclei in the system to LAMMPS for force evaluations using the same socket.

3. Let's run an i-PI+LAMMPS Molecular Dynamics simulation! Open a terminal at the current directory and launch i-PI by typing

   ```
   $ i-pi input.xml
   ```

   At this point i-PI should start and parse the input file. At the bottom of the output on the screen it should say

   ```
   Created unix socket with address 300K-G.0
   @ForceField: Starting the polling thread main loop.
   ```

   This means i-PI has started properly, has created the UNIX socket, and is waiting for the communications from the clients that do force evaluations.

4. Now it is a good time to start LAMMPS. Open up a second terminal either manually or by typing `Ctrl+Shift+t` and start LAMMPS by entering the command

   ```
   $ lmp_ubuntu < in.lmp
   ```

   Then LAMMPS should start and dump out some outputs.

5. Now switch to the terminal where i-PI is running, notice that i-PI has built the connection with LAMMPS with the message

   ```
   @SOCKET:·· Client asked for connection from . Now hand-
       shaking.
   @SOCKET:·· Handshaking was successful. Added to the client
        list.
   ```

   and started the Molecular Dynamics simulation. It should also dump out information on the time cost of each MD step.

6. What we are going to do now is to kill LAMMPS. Simply switch to the terminal where LAMMPS is running and press `Ctrl+c`. Now look at whether i-PI is still running. Notice that although the evolution of MD is paused, i-PI itself does not die off but instead continue to run and wait for new client to take over. Now start LAMMPS again by typing

   ```
   $ lmp_ubuntu < in.lmp
   ```

   What happens to i-PI now?

7. What if one stops i-PI? Kill i-PI by typing `Ctrl+c` where it is running, or create a file named `EXIT` in the folder where i-PI is running (you can use the bash command `touch EXIT`). Watch how i-PI respond, and how LAMMPS reacts. Think about what are the advantages of a clean exit when a MD program stops unexpectedly.

8. Take a look at all the output files dumped by i-PI. You should have `gas-nvt-300k.out` that describe the system properties, `gas-nvt-300k.xc.xyz` that records the atomic trajectories, and `RESTART` that contains all the information to restart the simulation.

Exercise 2 Keywords, outputs, and units of i-PI

In this exercise we are going to familiarize ourselves with the important notations of i-PI input files. A good way to learn is via examples. So let's take a look at the input files in the folder `ex-2/cp2k`. The simulation system of choice here is one gas phase water molecule using NVT ensemble at 300K. We are going to use CP2K for force evaluations.

1. Let's take a close look at the units first, which is *the* trickiest part of simulation and analysis. All the units used internally by i-PI are atomic units, as given below.

Unit	Name	S.I. Value
Length	Bohr radius	5.2917721e-11 m
Time	N.A.	2.4188843e-17 s
Mass	Electron mass	9.1093819e-31 kg
Temperature	Hartree	315774.66 K
Energy	Hartree	4.3597438e-18 J
Pressure	N.A.	2.9421912e13 Pa

   By default, both input and output data are given in atomic units, but in most cases the default units can be overridden if one wishes so. For example, in `ex-2/cp2k/input.xml` we used

```
<properties stride='1' filename='out'>
 [ step, time{picosecond}, conserved, temperature{kelvin
    }, potential, pressure_cv{atmosphere} ]
</properties>
<trajectory filename='xc' stride='1'>x_centroid{angstrom
    }</trajectory>
```

   so that the time, temperature, pressure, and the trajectory will written out in the units of picosecond, Kelvin, atm, and Angstrom in the output files, respectively. Similarly, the units of the initialization files can be specified such as

```
<file mode="xyz" units="angstrom"> h2o.xyz </file>
<cell mode="abc" units="angstrom"> [ 8, 8, 8 ] </cell>
<velocities mode="thermal" units="kelvin"> 300</
    velocities>
```

   When using i-PI, you can play around with the units to facilitate the initialization process as well as subsequent analyses.

2. Now we will observe the format and structure of `input.xml`. The xml file consists of a set of hierarchically nested tags. There are three parts to an xml tag. Each tag is identified by a tag name, which specifies the class or variable that is being initialized. Between the opening and closing tags there is some data, which is used to specify the contents of a class object, or the value of a variable. Finally tags can have attributes, which are used

to specify how the tag should be interpreted. A xml tag has the following syntax:

```
<tag_name attribute_name=attribute_data>tag_data</tag_name
    >
```

For example, in the tag

```
    <thermostat mode="pile_l">
        <tau units="femtosecond">100</tau>
        <pile_lambda>0.1</pile_lambda>
    </thermostat>
```

the tag name `thermostat` indicates which part of the simulation options is being defined, `mode="pile_l"` specifies the type of thermostat in use, and `<tau units="femtosecond">100</tau>` is used to set the parameters used for this particular thermostat. Please browse around this `input.xml` to see if you can make sense of most of the attributes.

3. It is worth spending a bit more time to explain about the type of socket used here. For the communication between i-PI and client codes, both Internet and Unix domain sockets can be used: the latter allow for fast communication on a single node, whereas the former make it possible to run i-PI and the clients on different computers. Here we demostate how to the Internet domain sockets. In i-PI inout file we have

```
    <ffsocket mode='inet' name='cp2k'>
        <latency> 1.00000000e-02</latency>
        <slots>4</slots>
        <port> 20614 </port>
        <timeout> 6.00000000e+02</timeout>
        <address> localhost </address>
    </ffsocket>
```

and in CP2K input file `cp2k.in` we have the following

```
&MOTION

  &DRIVER
    HOST localhost
    PORT 20614
  &END DRIVER
&END MOTION
```

The above means that an Internet domain socket with port number 20614 is used for the communication. The port number is an integer between 1 and 32277 used to distinguish between all the different sockets open on a particular host. As many of the lower numbers are protected for use in important system processes or Internet communication, it is generally advisable to only use numbers in the range 1025-32277 for simulations.

4. Now let's see whether the Internet domain socket can indeed connect. Type the following to start i-PI and CP2K:

```
$ i-pi input.xml &> log.ipi &
$ cp2k.popt -o h2o.out cp2k.in &
```

Does it work?

Exercise 3 Benchmark of quantum effects in a water molecule

In this exercise we will perform a series of PIMD calculations to observe and benchmark nuclear quantum effects in a molecular system. Our system of choice will be a single molecule of water in vacuum so that simulations don't take too much time and memory. We will be sampling an NVT ensemble using a q-TIP4P forcefield implemented within Lammps. We will look at the change in the kinetic and potential energy and the Oxygen–Hydrogen pair distribution function as we simulate in the quantum regime.

1. Look at the i-PI input file in `n.01/input.xml`. It is an i-PI input for a molecular dynamics simulations. Observe the properties and trajectory files that are specified in the <output></output> section. From the previous exercise, it should be clear that the quantum kinetic and potential energy will be printed out every 4 MD steps as columns 4 and 5 in the `$prefix.out` file. To compute the Oxygen–Hydrogen pair distribution function we shall also print out the positions of the atoms every 40 MD steps int the `$prefix.pos_0` file.
2. Observe the <initialize></initialize> section. We will be initialzing from a checkpoint file `start.chk` which is basically a RESTART file of an long equilibration run. From the previous exercise, it should be clear that the option `nbeads` should be 1 for a classical simulation.

```
<initialize nbeads='1'>
<file mode='chk' > init.chk </file>
</initialize>
```

3. Observe the <motion></motion> section. The tags within the section should be verbose enough to help you interpret their meaning. We will be using a "pile_l" thermostat which for the case of a classical simulation is a white noise Langevin thermostat. So that we are consistent with the rest of the PIMD runs, we will use an over-conservative time step of $0.25 fs$.
4. To run the simulation launch i-PI and lammps in background. It is advisable to keep separate logs as useful information can be extracted from them.

```
$ i-pi input.xml &> log.ipi &
$ lmp_ubuntu < in.lmp &> log.lmp &
```

To compute the average values of the observable use the program called `autocorr` which takes a time series as an input and evaluates its autocorrelation function. In addition to that it also computes the average and the associated error. Assuming that `$prefix` is the prefix used in i-PI input

```
$ awk '!/#/{ print $6}' $prefix.out | head | autocorr -
    maxlag 1
```

5. To compute the Oxygen–Hydrogen pair correlation function we suggest you to use `trajworks` and follow the syntax as shown.

```
$ cat $prefix.pos_0.pdb | trajworks -ipdb -vbox -gr -gr1 O
     -gr2 H -grmax 2 -hwin triangle -hwinfac 5·-> gOO.data
```

6. Now go back to the previous folder and then to folder called `n.32`. Try to run a PIMD simulation for 32 beads. You will find an `init.chk` file, containing an already equilibrated geometry of the extended system of 32 replicas of our system. Copy the i-pi and lammps inputs from `n.01`.
7. Make the modifications as indicated by the comments in the xml snippet.

```
<!-- change nbeads to 32-->
<initialize nbeads='1'>
<file mode='chk' units='angstrom'> init.chk </file>
</initialize>

<ffsocket mode='unix' pbc='false' name='driver'>
<!-- change the addess to driver.32-->
<address>driver.01</address>
<port>31400</port>
<latency>0.001</latency> <timeout>400</timeout>
</ffsocket>
```

8. Do not forget to change the address in the lammps input file. The comments in the following snippet should guide you.

```
neighbor 2.0 bin
timestep 0.00025
# replace driver.01 with driver.32
fix 1 all ipi driver.01 32346 unix
run 100000000
```

9. Similarly go to folders `n.16`, `n.08`, `n.04` and `n.02` and launch PIMD simulations for number of beads indicated by the name of the folders. Compute the pair correlation function and the energy for all the simulations using `autocorr` and `trajworks`.
10. If you prefer saving time you could use a scripts provided called `extract-pot.bash`, `extract-kin.bash` and `extract-gOH.bash` present in parent directory. These compute the potential energy, kinetic energy and the pair correlation function respectively.

```
$ pwd
/home/pimd/Desktop/PIMD TUTORIAL/day-1/ex-3
$ ls *.bash
extract-gOH.bash extract-kin.bash extract-pot.bash
```

11. Executing `extract-pot.bash` and `extract-kin.bash` will generate the average value and the error associated with the respective observable tabulated against the number of beads. The units and the reference values will be mentioned in the header.

```
$ pwd
/home/pimd/Desktop/PIMD TUTORIAL/day-1/ex-3
$ bash extract-pot.bash > pot.data
$ cat pot.data
#nbeads avg error
 1 4.822881e-02 5.040868e-03
 2 7.203741e-02 4.090793e-03
 4 1.453003e-01 4.452056e-03
 8 2.191862e-01 4.346367e-03
16 2.533971e-01 2.726154e-03
32 2.812411e-01 1.435196e-02
64 3.046206e-01 7.312336e-03
```

To plot the average estimates with errors bars launch `gnuplot` and type the following

```
p 'pot.data' us 1:2:3 w errorl
```

12. Executing `extract-gOH.bash` and `extract-kin.bash` will a file called gOH.data inside each of the sub directories `n.*`.

```
$ pwd
/home/pimd/Desktop/PIMD TUTORIAL/day-1/ex-3
$ bash extract-gOH.bash
$ ls */gOH.data
n.01/gOH.data n.02/gOH.data n.04/gOH.data n.08/gOH.data n
    .16/gOH.data n.32/gOH.data n.64/gOH.data
```

We suggest you to plot each of the pair distribution functions and see how they converge with number of beads. Launch `gnuplot` and type the following to

```
p 'n.01/gOH.data' us 1:2 w l, 'n.02/gOH.data' us 1:2 w l,
    'n.04/gOH.data' us 1:2 w l, 'n.08/gOH.data' us 1:2 w l
    , 'n.16/gOH.data' us 1:2 w l, 'n.32/gOH.data' us 1:2 w
     l, 'n.64/gOH.data' us 1:2 w l
```

Exercise 4 PIMD in the strong quantum regime: gas phase Methanium

Having benchmarked nuclear quantum effects in a water molecule we will study a CH_5^+ at 100 K where quantum effects such as tunneling and zero point fluctuations become strong. For a water molecule at 300 K $32beads$ were sufficient to accommodate NQEs, however in this case we will use 128 beads since at low temperature quantum effects are stronger. In other words, this simulation is 128 times

more computationally demanding than a MD. We shall be sampling a NVT ensemble and will look the proton delocalization by computing the Hydrogen–Hydrogen pair distribution function. For the purpose of comparison we will also run a cheap MD.

1. Move to the directory `ex-4/` and carefully observe thei-PI input files `n.001/input.xml` `n.128/input.xml`. You will be initializing form an already thermalized restart file called `init.chk`. We shall also be printing out the trajectory every $1fs$ so that we have enough statistics to compute the Hydrogen–Hydrogen pair distribution function. Run the classical and quantum simulations in their respective directories.

   ```
   $ i-pi input.xml &> log.ipi &
   $ cp2k_serial < in.cp2k &> log.cp2k &
   ```

2. To compute the Hydrogen–Hydrogen pair correlation function we suggest you to use `trajworks` and follow the syntax as shown.

   ```
   $ cat $prefix.pos_* | trajworks -ipdb -vbox -gr -gr1 H -
       gr2 H -grmax 3 -hwin triangle -hwinfac 5·-> gHH.data
   ```

3. You can use `gnuplot` to observe the difference int he pair distributions functions with and without PIMD. Launch gnuplot by typing `gnuplot` on the terminal and execute the following statement

   ```
   p 'n.128/gHH.data' us 1:2 w l, 'n.001/gHH.data' us 1:2 w l
   ```