An introduction to i-PI

Riccardo Petraglia, Michele Ceriotti

January 2017

In this set of exercises we will learn about the basics of i-PI and its use jointly with QUANTUM ESPRESSO.
As may be already clear, i-PI is not meant to compute forces. Thus, an external engine is needed. Once the external engine is set, i-PI takes care of the generation of the MD trajectory. This approach is extremely flexible. For example different kind of forces can be mixed within i-PI permitting the use of forces coming from different sources: correcting terms from different software (`PWscf` + external dispersion correction) or multiple time step approaches with expensive/cheap forces computations (see the Ring Polymer Contraction method).

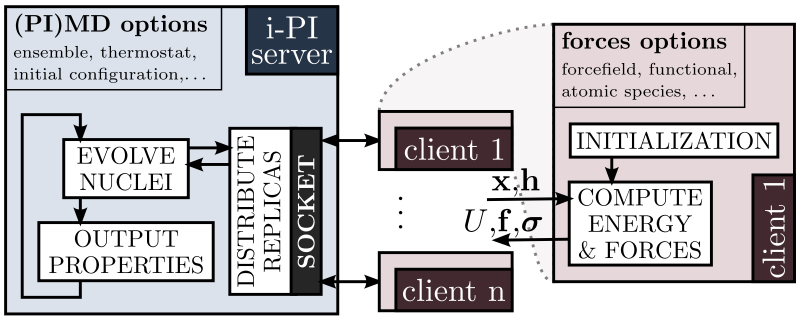

Figure 1: i-PI communicates with the force engine using sockets in a client/server fashion where i-PI behaves as the server. The engine connects to the server and provide i-PI the data it asks for.

The following exercises are created to increase the confidence of the user with the i-PI workflow and to clarify the server/client approach on which the i-PI-engine communication protocol is based (see fig. 1). Moreover, you will understand how to set-up a simulation using the PIGLET thermostat to improve the Path Integral Molecular Dynamics efficiency.

Exercise 1 Molecular Dynamics: a client/server approach

In this exercise, we will perform a simple Molecular Dynamics (MD) calculation using `PWscf` as engine. Since the computation of QM forces is expensive, here

we use a single molecule of water in gas phase. Moreover, the `PWscf` input is prepared sacrificing its accuracy to run faster.
The simulation will be performed using the NVT ensemble at 300K. We will also record the kinetic and potential energy and the conserved quantity of the system and the positions of the atoms.

1. Read the `PWscf` input file `ex-1/pw.in`. Is there any difference in respect of a plain single point computation? How is told to `pw.x` where to find the i-PI server?

   The input of `pw.x` is a valid input to run a single point computation. To make `pw.x` aware of i-PI, you should specify an option when calling the software (see section 3.6 of PWscf user's manual).

   ```
   pw.x --ipi host:port < pw.in
   ```

   The forces are then computed by `pw.x` at the level of electron theory specified in the input and passed to i-PI thought the socket interface.
2. Open the i-PI input file `ex-1/ipi.xml`. Specifying in the input which is the address of the socket where i-PI will wait for a `pw.x` instance is essential. You have two possibilities: using a UNIX-domain or an INET-domain socket. The former is faster but the server and the client must run on the same machine. The latter allows communication through the "internet" so that i-PI and the "force calculator" can run on a different computer.
   The following steps tell i-PI to wait for a client connecting on the UNIX socket called `single-water-qe`. Look for the following snippet in the `ex-1/ipi.xml`:

   ```
   <ffsocket name="..." mode="unix">
   <address>...</address>
   </ffsocket>
   ```

   Edit the file replacing the dots with appropriate values. Any string would be fine but, if you want to be consistent with the rest of the tutorial, you should use the following:

`name`	$\longrightarrow$	`pw-forces`
`address`	$\longrightarrow$	`single-water-qe`

   Pay attention that attribute's values must be within single- or double-quotes!

   The `name` attribute in the `ffsocket` node assigns a reference to the forcefield within the i-PI input. This reference will be used to specify the origin of the force within the `system` node.
3. Introduce a `<force>` node that references to the previously defined `<ffsocket>`. i-PI can manage different simulations at the same time. Each simulation is defined in the `<system>` node. Each `<system>` can also use forces coming

from different calculators. Since this is an introductory course, we are going to use a single system with a single force. In the file ex-1/ipi.xml look for the following snippet:

```
<forces>
<force forcefield='...'/>
</forces>
```

Replace the three dots with the name of the forcefield you want to use to compute the forces in this system. If you used the names suggested in the tutorial, it should be pw-forces.

4. We are now ready to run the simulation.

   The first thing to do is to start the i-PI server. Use the following command when in the folder ex-1:

   ```
   i-pi ipi.xml
   ```

   If everything is fine, the shell will show some messages and will stop with the following string:

   ```
   Created unix socket with address single-water-qe
   @ForceField: Starting the polling thread main loop
   ```

   this means that the server started correctly and is waiting for a client that computes forces.
   To start pw.x as an i-PI client open a new terminal, enter the folder ex-1 and create a new directory. Enter the just created folder and run the following command after replacing the unix_address with the actual address where i-PI is waiting (if followed tutorial suggestions, this should be pw-forces):

   ```
   pw.x --ipi unix_address:UNIX < pw.in
   ```

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

   ```
    @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. Try to kill the pw.x instance. Simply switch to the terminal where pw.x 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 pw.x again using the previous command. What happens to i-PI now? ...Nice eh?

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 responds, and how `pw.x` reacts. Think about what are the advantages of a clean exit when a MD program stops unexpectedly.
8. Checking the `<initialization>` node in the i-PI input, we see the attribute `nbeads=4`. This means that we are running a Path Integral Molecular Dynamics (PIMD) simulation with 4 beads. Easy, isn't it?. This explains why i-PI is producing so many xyz files as output. There is an output for each bead and a fifth file containing the trajectory of the centroid. You can open them all together with the following command:

   ```
   vmd -m example*.xyz
   ```

   You can also look at the `ex-1/example.out` file. This is the result of the `<property>` node. The header of the file summarizes what is printed in each column. You can create a graph using the following command:

   ```
   plot example.out -1,3
   ```

   The `-1,3` keyword tells to the script to use the column 1 as x-axis and the column 3 as y-axis. You can discover more option simply writing `plot` and pressing enter.

Before starting with the next exercise, take a few minutes to explore all the other keyword in the i-PI input. If you are familiar with other MD software, you should find the input quite self-explaining. Try to focus also on the `<output>` node and understand why i-PI is printing all those files.
When you are tired of looking at a single molecule of water, go to the next exercise. MD simulations are quite computationally expensive and performing simulation with *ab-initio* force evaluation would require much more time than what you can afford during this tutorial. This is why, from now on, the code LAMMPS will be used instead of QUANTUM ESPRESSO. All the input for QUANTUM ESPRESSO will anyway be present in the exercise `run` folders.

Exercise 2 Liquid water with the *PIGLET* thermostat

In this exercise you are going to learn how to perform a PIMD simulation using the PIGLET thermostat to improve the efficiency of the sampling.

1. Open the i-PI input file at `ex-2/ipi.xml` and localize the node describing the thermostat. The attribute `mode` of the thermostat is set to `langevin`. The PIGLET thermostat requires some parameters that can be downloaded from the following website `https://epfl-cosmo.github.io/gle4md/index.html?page=matrix`. To obtain the right parameters do the following:
   1. Select `PIGLET` as **GLE type**;
   2. Set the number of beads **N. beads** to `6`;

3. The **Par. set** should be `Ns=8`, $\hbar\omega/kT = 20$, `PIGLET`
4. To get the output in the i-PI input form, select i-PI in the **Output format**.
5. All the other parameters are left to the default value.

In this context, the `Ns` are the number of degrees of freedom of the bath, ω_{max} and `centroid` are the maximum and the window frequencies for which the sampling is enhanced. **Target T** is the temperature of the simulation and $\hbar\omega/KT$ is the ratio between vibrational energy and average kinetic energy.
Before proceeding further, do a copy of the entire directory `ex-2` and call it `ex-2-piglet`. Enter this latter and replace the langevin thermostat (all the `thermostat` node must be substituted) into the file `ex-2-piglet/ipi.xml` with the code obtained from the website.

2. Change the `verbosity` value from `medium` to `high` and start i-PI with the following command:

```
i-pi ipi.xml
```

from the `ex-2-piglet` folder.
At this point you have two choices for the force provider: you start `PWscf` or LAMMPS. Since this simulation contains 96 atoms, we warmly suggest using LAMMPS in this test by running the following command:

```
lmp_serial -i in.water
```

in a different shell. While the simulation is running, in order to understand the i-PI machinery better, you could prepare a `pw.x` input for this example starting from the `pw.x` input provided in the folder `ex-1/pw.in`. Try also to run it with i-PI to be sure it's working.
What are the lines inside the LAMMPS input which set the communication with i-PI?

3. Stop LAMMS (press `Ctrl+c` in the terminal where it has been ran). i-PI is now waiting for a new connection. Restart LAMMPS with the following command:

```
lmp_serial -i in.water > lmp.out.1 &
```

The shell should still present you the prompt. Run the same command a second time replacing the last "1" with a "2". What is happening in the i-PI terminal? You can see that now i-PI prints out much more information. Can you understand what they are? How many instances of LAMMPS could be used to maximize the performances?

With this second exercise we performed a much more interesting simulation using some advanced method to increase the efficiency of the PIMD statistical sampling. Before moving to the next exercise, we strongly suggest to run the same simulation using a white noise Langevin thermostat and a single bead (classical dynamics). Let both run while working on the next exercise. We will come back to these at the end of the tutorial to analyze some results.

At this point, you should know i-PI well enough to proceed with the next simulation without much help. Pay attention to the fact that only one instance of i-PI can use a given socket at time. If you want to run two i-PI server at the same time, make sure they have a different `<address>` or, in the case of INET domain sockets, at least different `<port>`. At the end, you can look at the differences between the two trajectories.

Exercise 3 PIMD-NPT simulation of ice

The goal of this exercise is to run a PIMD simulation in the NPT ensemble. The i-PI input provided with the file `ex-3/ipi.xml` is a valid input for a PIMD-NVT simulation. The idea is to transform it in a valid input for a PIMD-NPT simulation.

1. Take a look at the input provided and identify the nodes responsible for the task i-PI will perform. `motion` is the most important node in this case. Look at the following snippet:

```
<motion mode='dynamics'>
<dynamics mode='nvt'>
<thermostat mode='langevin'>
<tau units='femtosecond'> 100 </tau>
</thermostat>
<timestep units='femtosecond'> 0.25 </timestep>
</dynamics>
</motion>
```

   This section of the input tells i-PI that the computation we want to perform is a molecular `dynamics`, in the `nvt` ensemble using a `langevin thermostat` and a `timestep` of 0.25 femtosecond.
   There is also another important section:

```
<ensemble>
<temperature units='kelvin'> 200 </temperature>
</ensemble>
```

   This node of the input define all the ensemble properties. In the case you want to perform an NPT simulation, the pressure must be defined within the `ensemble` node.

2. Add a barostat. You can follow the user guide to understand how to set-up the barostat. Anyway, the following snippet provide a general setup which is suitable for this case.

```
<barostat mode='isotropic'>
<tau units='femtosecond'> 200</tau>
<thermostat mode='langevin'>
<tau units='femtosecond'> 100</tau>
</thermostat>
<h0> [ 25.6156, 0, 0, 0, 29.5783, 0, 0, 0, 27.8867 ]</h0>
```

```
</barostat>
```

Where is the place for this node? The `barostat` is a characteristic of the `dynamics` itself. Then it must be specified within that node. Copy the snippet aforementioned and paste just before the `<timestep>` node.

3. What if we want to perform an NPT simulation with an anisotropic barostat? This kind of simulations need an extension of the NPT ensemble, called NST (constant-temperature, constant-stress). Thus, in i-PI set the attribute `mode` of the `dynamic` node to `nst`. This means also that the `<ensemble>` node must contain a stress tensor instead of an isotropic pressure. To simulate our system with an anisotropic cell at atmospheric pressure, use the following snippet inside the `<ensamble>` node:

   ```
   <stress units='megapascal'> [10, 0, 0, 0, 10, 0, 0, 0, 10]
        </stress>
   ```

   The stress is a 3x3 tensor. To simplify user's life, this kind of tensors are flatten in the i-PIinput.
   At this point the input is ready to perform a simulation with at "constant anisotropic pressure".
4. What does the list `h0` contains? Why do we need a thermostat within the barostat?
   The list of numbers is a reference cell for the Parrinello-Rahman barostat (which is the one implemented in i-PI). In practice, those numbers are the dimension of the cell that must be written into the xyz input file (`ex3-1/init.xyz`) or in the pdb file (take as an example the one at `ex1/h2o-init.pdb`). The coordinate files also contains the units of dimension of the cell and the positions (`cell{angstrom}` and `position{angstrom}`).
5. Following the same procedure used in the last exercise, start this simulation.

Exercise 4 Radial distribution function comparison

As a last exercise, you should compare the radial distribution function obtained from the two simulation of exercise 2.

1. Compute radial distribution functions for the different simulations. You can use any tool of your liking, but we suggest the `trajworks` post-processing tool that is part of the `toolbox` library. Run the following command in the two folders used during the exercise 2: `ex-2` and `ex-2-piglet`.

   ```
   $read_box <(tail -q -n +100 example.pos_*.xyz) > box.dat%$
   $tail -q -n +100 example.pos_*.xyz | trajworks -ixyz \
   ·· ·· ·· ··-box box.dat -gr -gr1 O -gr2 O -grmax 5 -grbins
       250 -hwin \
   ·· ·· ·· ··triangle··-hwinfac 2 > gOO.dat%$
   ```

   To visualize the result you can use the `plot` command from `ex-2-piglet` folder:

   ```
   plot gOO.dat ../ex-2-langevin/gOO.dat -1,2
   ```

Also compute the O–H radial distribution function. Do you see any difference? Can you explain why?

We hope this tutorial made you familiar with the use of i-PI. You can find more tutorials and more information (as well as a link to the last development and stable versions at `http://ipi-code.org`.