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).

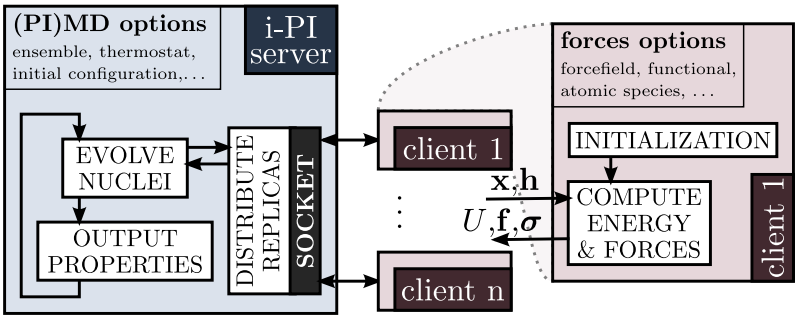

Figure 1: i-PI communicates with the force engine using sockets in a client/server fashion where i-PI behaves as the server. The engine connects to the server and provides to i-PI the data it asks for.

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.
If you are using the VirtualBox image, all exercises are in the sub-folder of `/i-PI tutorial/exercises`. The folders with the `-run` suffix contain the files resulting from the tutorials. If you have doubts on how to proceed, having a look at those files can be helpful.

Exercise 1 PIMD: a client/server approach

In this exercise, we will perform a simple Path Integral Molecular Dynamics (PIMD) 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 the accuracy to run faster – so do not use it as a template for any production run.
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 compared to a plain single point computation? How is pw.x told 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 argument when starting the software (see section 3.6 of PWscf user's manual).[1]

   ```
   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. Copy the i-PI input file ex-1/ipi-template.xml in ex-1/ipi.xml and open the latter. Specifying in the input 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(s) 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 computers within a cluster, or even (firewalls and network settings permitting.
   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    ⟶    pw-forces
   address ⟶  single-water-qe
   ```

[1] Do not run the following command now! You need to have first set up and launched the i-PI server, as will be explained further down.

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. If we want to use "INET" sockets, beside the `<address>` node we have to provide a `<port>` 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 this new 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 `single-water-qe`):

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

   When we use a UNIX socket, there is no need for specifying a port. The `port` field, as shown in the first point of this exercise, is then replaced by the `UNIX` string that tells `pw.x` to use a UNIX socket.

Optional
If you have time, after finished the tutorial, you can try using an INET socket.

Define the address of the machine where i-PI is running as <address> of the <ffsocket> node. If also the client will run on the same machine, you can use localhost. Otherwise, if the connection between i-PI and the client uses internet, use as <address> the actual IP address of the machine. The port can be any integer in the range 15000-45000. When you start the client, remember to modify the address and the port to match the ones used within the i-PI input.

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 and about which pw.x instance is used (you can have more than one) from each replica (bead).

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 it continues 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. When i-PI dies cleanly, it generates a RESTART file on the working directory. You can use that RESTART as a valid input for i-PI. If look inside the file you can find all the nodes that are also defined in the initial input. Beside them, you also have many nodes to specify the status of the various quantities when i-PI terminated.

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. 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, stop the simulation (exiting from i-PI as shown above) and take a few minutes to explore all the other keywords in the i-PI input. If you are familiar with pother MD software, you should find the input quite self-explanatory. 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 you have during this tutorial. This is why, from now on, the code LAMMPS will be used. All the input for QUANTUM ESPRESSO will be present anyway 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. To start the exercise, perform the following steps:
   - Copy the folder `ex-2` in `ex-2-piglet`.
   - Enter the new folder, copy the file `ipi-template.xml` to `ipi.xml`.
   - Open the new file and locate the node describing the thermostat (it is called `<thermostat>`).

   The attribute `mode` of the thermostat is set to `piglet`. The PIGLET thermostat requires some parameters that can be obtained from the following website `https://epfl-cosmo.github.io/gle4md/index.html?page=matrix`. The parameters that have already been included in the input – which are correct for simulations of aqueous systems at room temperature – can be obtained setting the following parameters:
   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.
2. Use the following command to run the PIGLET/PIMD simulation with i-PI:

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

At this point you have two choices for the force provider: you may 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. Keep this simulation running until the end of the tutorial.

3. Run an MD simulation using a White Noise Langevin (WNL) thermostat. Make a copy of the folder `ex-2` and call it `ex-2-langevin`. Enter this latter, make a copy of `ipi-template.xml` to `ipi.xml` and open this latter. Replace the PIGLET thermostat (all the `<thermostat>` node: from `<thermostat>` to `</thermostat>`) with the following snippet:[2]

```
<thermostat mode='langevin'>
<tau units='femtosecond'>100</tau>
</thermostat>
```

Moreover, since we want to run a classical MD (not PIMD), set the value of the attribute `nbeads` into the `<initialize>` node to 1.
Since each socket must be unique, the value of the `<address>` node must be changed: this one is already in use by the previous i-PI instance. Replace the `<address>` value with `ex2-2` and start the i-PI server with the usual command:

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

Before starting the LAMMPS instance we need to change the address inside the LAMMPS input file. Open a new terminal, move to the `ex-2-langevin` folder and open the file `in.water`. Locate the line relative to i-PI. You can find it toward the end of the file. Replace the value of the address (`ex2-1`) with the new one: `ex2-2`. Now start the LAMMPS simulation with the already known command:

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

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 (remember that you will have to change the socket address if it is already used).
Do not stop the simulations of exercise 2!

With this second exercise we performed a much more interesting simulation using some advanced method to increase the efficiency of the PIMD statistical sampling.

[2] Note that a white-noise Langevin thermostat is generally not a good choice to simulate a liquid. You could try a global stochastic velocity rescaling (`mode="svr"`) or fetch one of the optimal-sampling GLE thermostats from `https://epfl-cosmo.github.io/gle4md/index.html?page=matrix`

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 servers at the same time, make sure they have a different `<address>` or, in the case of INET domain sockets, at least different `<port>` values.

Exercise 3 PIMD-NPT simulation of ice

The goal of this exercise is to run a PIMD simulation in the NST ensemble (constant temperature and stress). The i-PI input provided with the file `ex-3/ipi-template.xml` is a valid input for a PIMD-NPT simulation. Make a copy of the template file to `ex-3/ipi.xml`. The idea is to start from the latter file to prepare in a valid input for a PIMD-NST simulation.

1. Take a look at the input provided and identify the nodes responsible for the task i-PI will perform. The attribute `mode` of the node `motion`, in our case, is set to `dynamics`. This attribute is the responsible in defining the task i-PI will attend. All the data contained into the `<dynamics>` node define the characteristics of the simulation: thermostat, barostat, timestep.
   There is also another important section:

   ```
   <ensemble>
   <temperature units='kelvin'> 200 </temperature>
   <pressure units='megapascal'> 0.1 </pressure>
   </ensemble>
   ```

   This node of the input define all the ensemble properties. In the case you want to perform an NST simulation, replace the `<pressure>` node with the `<stress>` one.

2. Adapt the `<barostat>` to run an NST simulation. The main difference between NPT and NST simulation is that the latter use an anisotropic barostat. You can see that the `<barostat>` node has an attribute `mode` that is set to `isotropic`. Set it to `anisotropic`. This means that also 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 `<ensemble>` node:

   ```
   <stress units='megapascal'> [0.1, 0, 0, 0, 0.1, 0, 0, 0,
       0.1] </stress>
   ```

   The stress is a 3x3 tensor. To simplify user's life, this kind of tensors are flattened in the i-PI input.
   At this point the input is ready to perform a simulation at "constant stress".

3. What does the list `h0` contains in the `<barostat>` node? Why do we need a thermostat within the barostat?

The list of numbers is a reference cell for the Parrinello-Rahman barostat (a derivative of which is implemented in i-PI). In practice, those numbers are the dimension of the cell that describes the "average" size of the system. In general results are fairly insensitive to the precise value. The coordinate files also contain the units of dimension of the cell and positions (`cell{angstrom}` and `position{angstrom}`).

4. Following the same procedure used in the last exercise, start this simulation. Pay attention to use different sockets from the ones used in exercise 2.
5. **This is optional, do it only if you have enough time!**
   Stop i-PI (press `Ctrl+c` in the terminal where it has been ran). Edit the `ipi.xml` file and change the verbosity (fist line) to `high`. Restart i-PI with the usual command. Once i-PI is waiting for a new connection, start LAMMPS with the following command:

   ```
   lmp_serial -i in.ice > 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?[3]

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. The following command is a facade to the `trajworks` tool that speed up your work. Run the command from the two folders used during the exercise 2: `ex-2-langevin` and `ex-2-piglet`.

   ```
   $make_gofr O O
   ```

   This produce a file called gOO.dat inside the folder with the $g(r)$ between oxygen atoms. Do the same to compute the H-O radial distribution function. 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 visualize the $g(r)$ for 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`.

[3] Note that i-PI is designed with ab initio simulations in mind, so the parallel scaling when using inexpensive empirical forcefields will not be perfect – due to the communication overhead and the intrinsic limitations of Python.