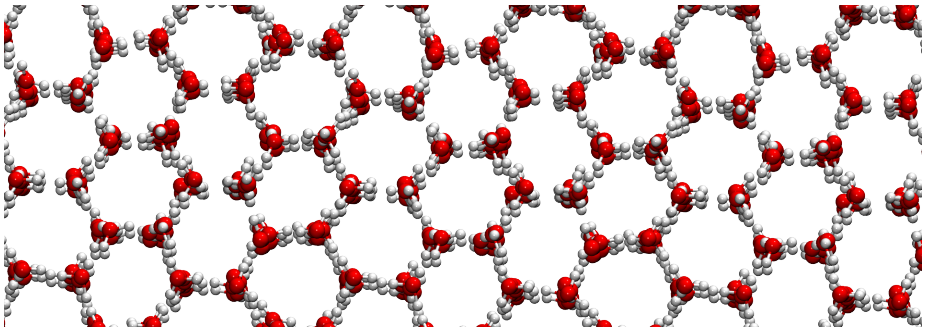
# Path integral approximations to real time correlations

Mariana Rossi

May 2016



In this exercise we will perform simulations of Ice Ih at 200 K, using a simple empirical force field model based on TIP4P-like point charges. We will use this model to look at different path integral approximations to real time correlations, calculate imaginary time correlations from PI, and look at the relationship between them. The ice box contains 96 water molecules. All files necessary for the exercises below can be found in the `day-3` folder.

## Exercise 1 Velocity and dipole real time correlation functions

In this first exercise, we will calculate time correlation functions with path integral based methods, namely centroid molecular dynamics (CMD) [1], ring polymer molecular dynamics (RPMD) [2], and thermostated ring polymer molecular dynamics (TRPMD)[3]. All these methods have been explained in the lectures, and we encourage you to look at the respective references for in more details. We note that for the TIP4P-style of charges, the dipole moment is a linear function of the coordinates, and therefore the dipole estimator in CMD, RPMD, and TRPMD is always the same and can be calculated from the centroid positions (or velocities, in the case of its derivative). Due to time constraints we will only perform short (10 ps) simulations, and you'll be provided extensive outputs for further analysis in the exercises that require it. Inputs for simulating different PI-based methods are also provided.

- **Vibrational spectra with classical nuclei (Time: 10 minutes)**
  We will start by calculating both dipole and velocity correlation functions with classical nuclei. In order to do that, we have provided pre-computed i-PI checkpoint files from a long thermalization run of Ice Ih at 200 K. Checkpoint files are a feature of i-PI that can be written through the keyword <checkpoint> in the <output> block of i-PI's input file, for which a syntax example would be:

  ```
  <checkpoint filename="restart" stride="2000" overwrite="
      false" />
  ```

  Inserting this keyword in a thermalization run provides a series of files that can be used to restart simulations from different points of the thermalization simulation (defined by the option `stride`). You can find checkpoint files from an NVT run with classical nuclei in the `ex-1/checkpoint-nvt` directory.

  In order to run the NVE trajectory needed to compute the correlation functions do:

  ```
  $ cd ex-1/nve
  $ vi nve.xml
  ```

  The file `nve.xml` is an example of input for NVE simulations. Take some time reading through it. For the current simulation, the most important quantity we will print out is the velocity of the atoms (here given by the centroid velocities since there is only one bead). You will notice that there are some fields marked with `xxx` in the input file. These should be *substituted by a number of your choice*, corresponding to the identity of the checkpoint file you want to choose. You can substitute `xxx` by a number by hand or use a command. For example, in order to create a new directory with all necessary input files, run:

  ```
  $ number=6
  $ mkdir dir_${number}
  $ sed "s/xxx/${number}/g" nve.xml > dir_${number}/input.
      xml
  $ cd dir_${number}
  $ cp ../../lammps_inputs/* .
  ```

  Now the directory `dir_$number` contains all input files necessary to start our simulation. In order to launch one instance of i-PI and LAMMPS, run:

  ```
  $ i-pi input.xml &> log.ipi &
  $ what=$(grep addr input.xml | sed "s/.*<address>//; s/<\/
      address>//")
  $ sed "s/addr/$what/" in.tip4pf | lmp_ubuntu &> log.lammps
       &
  ```

  The two last lines are there so as to inform LAMMPS (by changing its input), the address of the UNIX socket that is currently used by the i-PI server.

After approximately 10 minutes this simulation should be done and you can analyze the outputs. *Immediately after this simulation ends, launch the TRPMD simulation on the second item below before analyzing the outputs, so that it starts running in the background*.

Look at outputs using `gnuplot` and `vmd`. In order to create the dipole and velocity autocorrelations, we provide scripts in the `day-3/tools` folder that uses `trajworks` functionalities in order to perform fourier transforms and calculate auto correlation functions, called `post-vel` and `post-dpl`. We will actually calculate the dipole derivative auto correlation function, which already includes the $\omega^2$ factor that appears when relating the Kubo transform of the autocorrelation function to the standard canonical one.

Copy files to the folder where you performed the simulation and run the following commands:

```
$ cp ../../../tools/post* .
$ cp ../../../tools/charges.tip4pf .
$ sh post-dpl
$ sh post-vel
```

Visualize (either with `xmgrace` or `gnuplot`) the files `dpl.ft` (Fourier transform of the dipole auto correlation) and `vel.ft` (Fourier transform of the velocity auto correlation).

Analyze similarities and differences between them (remember this is a short simulation that cannot yield very converged correlation functions). In the folder `ex-1/precomputed-trajectories/nve/` you will find several longer precomputed and preanalyzed trajectories.

- **Vibrational spectra with quantum nuclei (Time: 1 hour)**
  Due to time constrains, we will only perform one TRPMD run with 48 beads. This simulation takes around one hour. For RPMD, more simulations would be needed for a reasonable spectra due to its poor ergodicity, and for (partially adiabatic) CMD, due to the much smaller time step required, the simulation would take 10 times longer.

  The input file for this simulation will be in the `ex-1/trpmd` folder. The checkpoints from a PIMD simulation that we will use to start the TRPMD one can be found in `ex-1/checkpoint-pimd`. The set of bash commands you will need to launch this simulation are the following (you can change the variable `number`):

```
$ cd ex-1/trpmd
$ number=6
$ mkdir dir_${number}
$ sed "s/xxx/${number}/g" trpmd.xml > dir_${number}/input.
    xml
$ cd dir_${number}
$ cp ../../lammps_inputs/* .
```

```
$ i-pi input.xml &> log.ipi &
$ what=$(grep addr input.xml | sed "s/.*<address>//; s/<\/
    address>//")
$ for i in `seq 1 8`; do sed "s/addr/$what/" in.tip4pf |
    lmp_ubuntu &> log.$i & done
```

Beyond the velocities, here we will print out also the positions and forces on each bead, which is necessary to build the imaginary time correlation functions that we will look at in Exercise 2.

Again after the simulation is done, from the simulation directory run:

```
$ cp ../../../tools/post* .
$ cp ../../../tools/charges.tip4pf .
$ sh post-dpl
$ sh post-vel
```

Visualize (either with `xmgrace` or `gnuplot`) the files `dpl.ft` and `vel.ft` and compare them with their classical counterparts that were calculated in the previous item. Which differences do you observe?

In the directories `ex-1/precomputed-trajectories/rpmd/`, `ex-1/precomputed-trajectories/trpmd/`, and `ex-1/precomputed-trajectories/cmd/` you'll find many precomputed and preprocessed trajectories from each different path integral based method.

Visualize and compare all Fourier transformed functions of dipole and velocities (.dpl and .vv), as computed from the different methods. Can you spot the spurious resonances in RPMD and the curvature problem in CMD? And do you see that the TRPMD peaks are much broader than the classical ones?

## Exercise 2 Velocity imaginary time correlation function

In this exercise, we will calculate an example of an imaginary time correlation function, namely the velocity-velocity imaginary time correlation. These quantities are exact withing the PI formalism and can be computed from PIMD, CMD, RPMD, and TRPMD, always yielding the same result since all these methods preserve ensemble. A direct estimator for the imaginary time correlation function on the imaginary time $i\lambda_j$, with $\lambda_j = j\beta_P\hbar$, $j = 0, ..., n-1$, and $\lambda_n = \lambda_0$ is

$$c_{vv}(i\lambda_j) = \sum_k^N \left[ \frac{3}{\beta_P m_k} - \frac{\omega_P}{P} \sum_i^{P-1} \langle (\mathbf{q}_i^{(k)} - \mathbf{q}_{i+1}^{(k)})(\mathbf{q}_{i+j}^{(k)} - \mathbf{q}_{i+j+1}^{(k)}) \rangle \right]. \quad (1)$$

For hints on the derivation of the equation above, check the appendix of Ref. [4]. This expression can be recast to a centroid virial estimator form that is

$$c_{vv}(i\lambda_j) = \sum_k^N \left[ \frac{3}{\beta m_k} - \frac{1}{P m_k} \sum_i^{P-1} \left\langle (\mathbf{q}_{i+j}^{(k)} - \bar{\mathbf{q}}) \frac{dV(\mathbf{q}_i^{(k)})}{d\mathbf{q}_i^{(k)}} \right\rangle \right], \quad (2)$$

where $\bar{\mathbf{q}}$ is the centroid position. This is what we will calculate from our simulation. A script to calculate this expression is available in `tools/posforce2imaginaryvv.py`. Run the following commands:

```
$ cd ex-2
$ cp -r ../ex-1/trpmd/dir_* .
$ cd dir_*
$ python ../../tools/posforce2imaginaryvv.py h2o-ice-trpmd 200
```

The file generated `h2o-ice-trpmd.imvacf` contains the velocity imaginary time correlation, where in the x axis is imaginary time in units of $\lambda_j\hbar/(P\beta\hbar)$. Plot this quantity. For the temperature we are simulating, we know the imaginary time correlation function through an imaginary time slice from $0$ to $\beta\hbar = 38.2$ femtoseconds.

A more converged imaginary time velocity correlation function is given in `ex-2/precomputed-imvacf/`.

## Exercise 3 Relationship between imaginary and real time correlations

A way to check the quality of the real time autocorrelation function approximated by the PI-based methods (like the ones calculated in Ex. 1) is to look at sum rules that connect the real time with the imaginary time correlations. For the velocity real and imaginary time correlations, the relationship is

$$c_{vv}(i\lambda\hbar) = \int_0^\infty f(\lambda,\omega)\tilde{F}_{vv}(\omega)d\omega \tag{3}$$

$$f(\lambda,\omega) = \frac{\beta\hbar\omega\cosh(\beta\hbar\omega/2 - \lambda\hbar\omega)}{2\pi\sinh(\beta\hbar\omega/2.)} \tag{4}$$

where $F_{vv}(\omega)$ is the Fourier transform of the Kubo transformed real time velocity correlation function.

In Ref. [5] one can also find the expression that relates the imaginary time square displacement and the real time velocity correlation function. Ref. [4] and references therein also point to derivations of these expressions.

By reconstructing $c_{vv}(i\lambda\hbar)$ from the vibrational spectra obtained with either TRPMD, RPMD, or CMD, and comparing to the exact $c_{vv}(i\lambda\hbar)$ obtained in Ex. 2, one can assess (even if indirectly) how large the errors are.

In order to reconstruct the imaginary time correlations from real time data we provide the script `tools/get_imacf_from_real.py`. The syntax is:

```
python get_imacf_from_real.py [filename] [temperature] [nbeads]
```

In the folder `ex-3/convergedspecs` you will find the files `trpmd.vel`, `rpmd.vel`, and `cmd.vel`. Type (the simulations provided here were run with 56 beads):

```
$ cd ex-3/convergedspecs
$ python ../../tools/get_imacf_from_real.py trpmd.vel 200 56 >
    trpmd.imvacf
```

```
$ python ../../tools/get_imacf_from_real.py rpmd.vel 200 56 >
    rpmd.imvacf
$ python ../../tools/get_imacf_from_real.py cmd.vel 200 56 >
    cmd.imvacf
```

Compare all the imaginary time correlations with the one provided in `ex-2/precomputed-imvacf/`. Which method yields the most accurate imaginary time correlation function?

## References


[1] Jianshu Cao and Gregory A Voth. The formulation of quantum statistical mechanics based on the Feynman path centroid density. IV. Algorithms for centroid molecular dynamics. *J. Chem. Phys.*, 101:6168–6183, 1994.

[2] I R Craig and D E Manolopoulos. Quantum statistics and classical mechanics: Real time correlation functions from ring polymer molecular dynamics. *J. Chem. Phys.*, 121:3368, 2004.

[3] Mariana Rossi, Michele Ceriotti, and David E Manolopoulos. How to remove the spurious resonances from ring polymer molecular dynamics. *J. Chem. Phys.*, 140:234116, 2014.

[4] Scott Habershon, Bastiaan J Braams, and David E Manolopoulos. Quantum mechanical correlation functions, maximum entropy analytic continuation, and ring polymer molecular dynamics. *J. Chem. Phys.*, 127:174108, 2007.

[5] Mark Tuckerman. *Statistical Mechanics and Molecular Simulations*. Oxford University Press, 2008.