Quantum Monte Carlo for Condensed Matter and Statistical Physics (M2)

TD session **1 and 2**

F. Mezzacapo, T. Roscilde

General goal and rules of the TD sessions: *The TD sessions are fully hands-on namely, in every TD session you are supposed to write and test different parts of a Path Integral Monte Carlo computer code designed to estimate the physical properties of quantum systems, in some selected cases. You can write the code alone or in a two-people team. You should choose a programming language (C++, Fortran, etc.) and be able to plot your results (using e.g., Gnuplot, the plotting utilities of Matlab, etc.). We assume that you have a reasonable familiarity with at least one programming language.*

Disclaimer: *If you wish, you can skip the analytical exercises proposed in Subsecs. 1.1 and 3.2, and directly use Eqs. (1), and (8). Exercises marked with **(F)** are facultative.*

1 The model

We will consider, as a starting point, the problem of a single quantum particle of mass m subjected to an external harmonic potential in one spatial dimension. The Hamiltonian of the system is $H = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial y^2} + \frac{1}{2}m\omega^2 y^2$, where ω is the angular frequency, $\hbar$ the reduced Planck constant and y the particle position.
By choosing $E_0 = \hbar\omega$ and $y_0 = \sqrt{\frac{\hbar}{m\omega}}$ as units of energy and length, respectively, one obtains the dimensionless form:

$$\mathcal{H} = \frac{1}{2}\frac{\partial^2}{\partial x^2} + v(x), \text{ with } v(x) = \frac{1}{2}x^2. \tag{1}$$

1.1

Verify Eq. (1).

In the following we first specialize some important formulas you have already seen during classes to our model in Eq. (1), then we will start going through the basic steps needed to implement a Path Integral Monte Carlo (PIMC) code. We will use our code to estimate the finite-temperature properties of our chosen model. It is worth mentioning that the implementation discussed here, and in the other TD sessions, can be adapted with minimal changes to tackle a very large class of (bosonic) problems in continuous space.

2 Useful formulas

The partition function of our system of interest, at a temperature $1/\beta$ is:

$$\mathcal{Z} = \int dx_0 \left\langle x_0 | e^{-\beta \mathcal{H}} | x_0 \right\rangle . \tag{2}$$

The integrand in Eq. (2) can be analytically evaluated due to the particular Hamiltonian in Eq. (1), however, given the purpose of these TD sessions we will use an approximation which is i) systematically improvable and ii) applicable regardless the details of the Hamiltonian. By inserting $M-1$ identities in Eq. (2) we obtain:

$$\mathcal{Z} = \int \prod_{k=0}^{M-1} dx_k \left\langle x_k | e^{-\tau \mathcal{H}} | x_{k+1} \right\rangle , \tag{3}$$

where $\tau = \beta/M$ is the imaginary time step, M the number of slices, $\boldsymbol{x} = x_0, \cdots, x_{M-1}$ is the path or world line and $x_M = x_0$. The k_{th} imaginary time propagator in Eq. (3) can be approximated as:

$$\left\langle x_k | e^{-\tau \mathcal{H}} | x_{k+1} \right\rangle \simeq (2\pi\tau)^{-\frac{1}{2}} e^{-\frac{(x_{k+1}-x_k)^2}{2\tau}} e^{-\tau v(x_k)}. \tag{4}$$

The above is known as primitive approximation: it is exact in the limit of small τ and accurate up to order τ^2. Defining

$$P(\boldsymbol{x}) = \frac{\prod_{k=0}^{M-1} \left\langle x_k | e^{-\tau \mathcal{H}} | x_{k+1} \right\rangle}{\mathcal{Z}} \simeq \frac{\prod_{k=0}^{M-1} (2\pi\tau)^{-\frac{1}{2}} e^{-\frac{(x_{k+1}-x_k)^2}{2\tau}} e^{-\tau v(x_k)}}{\mathcal{Z}}, \tag{5}$$

the expectation value of an operator $\mathcal{O}$ in terms of its estimator $O(\boldsymbol{x})$ is:

$$EXP[\mathcal{O}] = \int \mathcal{D}(\boldsymbol{x}) P(\boldsymbol{x}) O(\boldsymbol{x}), \text{ where } \mathcal{D}(\boldsymbol{x}) \equiv dx_0 dx_1, \cdots, dx_{M-1}. \quad (6)$$

For general models, Eq. (6) cannot be solved exactly. The PIMC method, in a nutshell, provides an efficient way of estimating the multidimensional integral above. Specifically, via PIMC one generates a large set of configurations $\boldsymbol{x}^1, \cdots, \boldsymbol{x}^{N_c}$ sampled from $P(\boldsymbol{x})$ and approximates Eq. (6) as:

$$EXP[\mathcal{O}] \simeq \frac{1}{N_c} \sum_{i=1}^{N_c} O(\boldsymbol{x}^i) = \langle O(\boldsymbol{x}) \rangle \,. \quad (7)$$

You may have noticed that in this section we have used two approximations: the primitive one for the propagators and that in Eq. (7). They are both "under control", becoming exact in the large M and N_c limit, respectively.

3 Setting up the code

Our PIMC code should be written in terms of a few adjustable parameters and arrays which will be progressively defined and consistently used in the TD sheets.

For example, **you may start using**:

β: dimensionless inverse temperature.

M: number of imaginary time slices, or beads.

$\boldsymbol{x}$: array of size M with the particle position at each slice (i.e., the world line).

N_c: number of configurations generated by our PIMC algorithm. A new configuration is generated starting from the current one, proposing a certain number of single-bead displacements (see below).

Also, you will need to use standard functions included in essentially all programming languages to extract uniformly or normally distributed random numbers.

3.1 Sampling the configuration space

The sampling strategy is one of the fundamental point of a PIMC code. As a first step you should assign the system initial configuration e.g., by uniformly extracting the position of our single particle for each bead in a certain range $[-XMAX, XMAX]$. Then you can modify the initial configuration by proposing M sequential single-bead updates as follows:

for($k = 0$; $k < M$; $k++$)
{
  Randomly select a bead j;
  Propose a new position for the selected bead:
    $x_j^{new} = \frac{x_{j+1}+x_{j-1}}{2} + g(\sqrt{\tau/2})$;
  Perform Metropolis acceptance/rejection test;
}
new configuration obtained;

The above pseudo-code where $g(\sqrt{\tau/2})$ is a random number distributed according to a gaussian with zero mean and standard deviation equal to $\sqrt{\tau/2}$ provides a simple "recipe" to generate a new global configuration by means of M sequential local updates. In the following, the procedure of attempting M sequential single-bead updates will be referred to as the proposition of a global update.

Within the scheme discussed above the displacement of a single bead j is accepted, for our model in Eq. (1), with probability

$$A(x_j \to x_j^{new}) = min\Big(1, e^{-\frac{\tau}{2}[(x_j^{new})^2-(x_j)^2]}\Big). \tag{8}$$

3.2

Verify Eq. (8).

3.3

After having implemented the sampling strategy introduced in this section you can do a first numerical exercise:
Initialize the system setting e.g., $x_i = 0 \; \forall \; i = 0, \cdots, M-1$. Propose several global updates (e.g., 5×10^5) and plot the final configuration for $\beta = 1.0$ and $\beta = 10^{-2}$, as well as the initial one (use $M = 10$). What do you observe?

[You may need to rigidly shift one of the two world lines to visualize both in the same spatial range].

4 Estimating observables I: Energy

We can now evaluate physical observables as averages over the sampled configurations of the corresponding estimators. For the total energy E of our particle the estimator reads:

$$\mathcal{E}(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M}\left[\frac{1}{2\tau} - \frac{1}{2}\left(\frac{x_{j+1}-x_j}{\tau}\right)^2 + v(x_j)\right]. \tag{9}$$

Hence,

$$E \simeq \langle \mathcal{E}(\boldsymbol{x})\rangle \tag{10}$$

Assuming to have generated a large number N_c^I of independent configurations and computed the corresponding large sample of uncorrelated $\mathcal{E}(\boldsymbol{x})$ values, the uncertainty on E can be estimated as:

$$\sigma_E \simeq \left(\frac{\langle \mathcal{E}(\boldsymbol{x})^2\rangle - \langle \mathcal{E}(\boldsymbol{x})\rangle^2}{N_c^I - 1}\right)^{1/2}. \tag{11}$$

The update strategy implemented in our code, however, relates configurations visited sequentially. As a result, the above formula used for the N_c configurations generated by our algorithm will generally underestimate σ_E. In order to overcome this problem one can divide the total number of sampled configurations in N_B blocks (with $N_B >> 1$) and compute the partial average E_B^i for each block (i.e., $\forall\ i = 1, \cdots, N_B$). If the block size (i.e. the number of configurations in each block) is large enough the E_B's are independent. Hence, they lead, used in the framework of Eq. (11), to the correct estimate of σ_E.

4.1 (F)

Set $\beta = 0.5$ and $M = 10$. Propose a large number $NTHERM$ of global updates to thermalize the system. Then, generate via global updates N_c configurations, compute $\mathcal{E}(\boldsymbol{x})$ for each configuration and write the $\mathcal{E}(\boldsymbol{x})$'s on

a file. Reblock your energy estimates defining $n_B = N_c/M_B$ partial averages and estimate

$$\sigma_E(M_B) = \left(\frac{\langle (E_B^i)^2 \rangle - \langle E_B^i \rangle^2}{N_c/M_B - 1} \right)^{1/2} \tag{12}$$

where $\langle \cdots \rangle$ here indicates average of the N_c/M_B partial averages. Plot $\sigma_E(M_B)$ as a function of M_B. You should see that this quantity first increases with increasing M_B and then flattens. You can take the plateau value as the uncertainty on E. In this exercise typical values you may use are $NTHERM = 10^5$, $N_c = 2^{18}$ and $M_B = 2^n$ with $n = 0, \cdots, 13$. What is your estimated energy? Is it compatible, tacking into account the errorbar, with the exact result? [$e^{ex}(\beta = 0.5) = 2.04149$]. How can you improve your estimate? [Note that in this exercise you need to run a single simulation and manipulate the output file to obtain $\sigma_E(M_B)$ for various choices of M_B].

4.2

In this numerical exercise we will explore the dependence of E on M. Set, for example, $\beta = 5.0$ and estimate the energy for a few M values. For each M you should run a separate simulation. To speed up the code avoiding to write very long files, you can build block averages "on the fly" by setting an a-priori value of M_B. During the simulation you can append on a file your block-averaged values of the energy. You can then estimate E (and σ_E) post-processing such a file. Note that, in this way, if needed, estimates can be further reblocked without rerunning the simulation. A typical value of M_B you can use as a starting point here is 2500. You can use a few values of M in the interval $[1, 50]$. What do you expect to obtain for $M = 1$? Plot $E(M)$ and compare your estimates with the exact result $E^{ex}(\beta = 5.0) = 0.50678$.

4.3

Estimate $E(\beta)$ for a few values of β, for example, in the interval $[0.5, 10]$. Plot $E(\beta)$ and compare it with the exact result:

$$E^{ex}(\beta) = \frac{1}{2\tanh(\beta/2)}. \tag{13}$$

In this exercise you can use $\tau = \beta/M = 0.1$.

5 Estimating observables II: Probability distributions

The probability distribution function $P(x)$ can be defined as:

$$P(x) = \mathcal{N} \sum_{j=1}^{M} \delta(x_j - x) \tag{14}$$

Operatively, you have to define a simulation box, i.e., a region of space large enough to contain the position of each bead during the simulation. Given our simulation box: $[-XMAX,\ XMAX]$ (where the value of $XMAX$ may depend on the simulation parameters), $P(x)$ is accumulated as an histogram as follows:

set the number of bins $NBIN$
$dx = 2XMAX/NBIN$;
for(int $k = 0$; $k < M$; $k++$)
{
  $auxbin =$ bin containing x_k;
  $P[auxbin] += 1$;
}
determine $\mathcal{N}$ via normalization (i.e., imposing $\sum_{ibin} P[ibin]dx = 1$)

Similarly, you can evaluate the probability distribution functions:

$$P'(x) = \mathcal{N}' \sum_{j=1}^{M} \delta((x_j - x_c) - x) \text{ with } x_c = \frac{1}{M} \sum_{j=1}^{M} x_j, \tag{15}$$

and

$$P''(x) = \mathcal{N}'' \delta(x_c - x) \tag{16}$$

which provide information about quantum and thermal fluctuations, respectively.

5.1 (F)

Evaluate $P(x)$, $P'(x)$ and $P''(x)$ for $\beta = 0.5$ and $\beta = 10$ [use $\tau = 0.1$]. What differences do you expect? Plot your results and compare your estimated

$P(x)$ with the exact expression:

$$P^{ex}(x,\beta) = \frac{2\sinh(\beta/2)}{\sqrt{2\pi\sinh(\beta)}} e^{-\frac{\cosh(\beta)-1}{\sinh(\beta)}x^2} \tag{17}$$

6 Generalizing the code

At this point you should have a working code for the case of a single particle. Such a code can be easily extended to the case of 2 and more distinguishable particles. To this goal you should:
1) Add an index to your configuration array $\boldsymbol{x}$: $x_j \to x_{i,j}$ where i, and j are the particle and slice index, respectively.
2) Use the straightforward generalization of the propagator in Eq. (4):

$$\left\langle \boldsymbol{x}_k | e^{-\tau\mathcal{H}_N} | \boldsymbol{x}_{k+1} \right\rangle \simeq (2\pi\tau)^{-\frac{N}{2}} e^{\sum_{i=1}^{N}\left[-\frac{\left(x_{i,k+1}-x_{i,k}\right)^2}{2\tau} - \tau v(x_{i,k})\right]} \tag{18}$$

where $\boldsymbol{x}_k \equiv \{x_{1,k}, \cdots, x_{N,k}\}$ and $\mathcal{H}_N = \sum_{i=1}^{N}\left(\frac{1}{2}\frac{\partial^2}{\partial x_i^2} + v(x_i)\right)$.
3)Define a global update as the propsition of $N \times M$ single-bead updates where the particle and the bead to update are both randomly chosen each time.
4) Use the generalized forms of Eqs. (9), (14-16) obtainable by simply taking an extra summation over the number of particles. The energy estimator will then read:

$$\mathcal{E}(\boldsymbol{x}) = \sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}\left[\frac{1}{2\tau} - \frac{1}{2}\left(\frac{x_{i,j+1}-x_{i,j}}{\tau}\right)^2 + v(x_{i,j})\right] \tag{19}$$

while, for example, the probability distribution in Eq. (15)

$$P'(x) = \mathcal{N}'\sum_{i=1}^{N}\sum_{j=1}^{M}\delta((x_{i,j}-x_{i,c})-x). \tag{20}$$

6.1

Compute E and $P(x)$ for $\beta = 0.5$ and $\tau = 0.1$. Compare the cases $N = 1$ and $N = 2$. What do you expect?

Next...

Identical bosons.