Discovering a new phase with Path Integral Monte Carlo

We will use the Worm Algorithm to explore the density-temperature phase diagram of Bose particles in two dimensions described by the model Hamiltonian $-\lambda\nabla^2+\sum_{i<j} v(r_{ij})$, where

$$v(r)=\begin{cases} a & \text{if } r<1; \\ 0 & \text{if } r>1. \end{cases}$$

We use atomic units and set $\lambda=1/2$ and the interaction strength $a=50$. The particles have spin zero.

Assignments:

Important notes:

You don't need to complete all assignments (1, 2 and some of 3 are enough)

Refer to the documentation appended below after the assignments; feel free to consult notes, internet, whatever (but not your colleagues...) and to ask questions.

For each assignment create a dedicated subfolder with the output files and a text file with your answers/comments as appropriate.

To submit your test, send an email to saveriomoroni@gmail.com with

- **a compressed file with the folder containing your test if you are using your laptop, or**
- **the name of the fcmxx account and the name of the folder containing your test if you are using an fcmxx account.**

1.
- Compile the fortran codes `worm.f` and `statforw.f` (if needed see below for instructions);
- Run a PIMC simulation with the provided input file `worm.in` (save the output on a file);
- Do you see any evidence that some initial blocks should be discarded for equilibration?
- Do you see autocorrelation between block averages?
- Do you need to adjust the size of the move to get an acceptance rate ~0.5 for the "wiggle" moves?
- Modify the input and run another simulation with statistical error bars ~twice as small;
- The PIMC method has a systematic but controllable error. Modify the input so as to reduce such an error and run another simulation. Compare the acceptance rate for the "wiggle" moves with the previous runs and comment.

2.
- The temperature and chemical potential in the provided input file correspond to a specific phase of the system; can you tell which phase? (collect in a dedicated subfolder the output files: the output of the simulation and any of gofr, sofk, nofr, restart.coord as you deem relevant).

3.
- The state of the system depends on the temperature and the chemical potential; modify the input file to characterize at least one phase different from that identified at 2. Note that equilibration times need not be the same for all phases; if needed, run longer runs and/or use "restart 1" and append the output of more runs to the output file. Use a dedicated subfolder to collect output files for each phase you find.

- Do you see any difference on the frequency of the Z-sector configurations between different phases? Tell what is (and/or modify) the relevant parameter in the input file to tune such a frequency.

4.
- If you found a “regular” solid phase at 3, you may want to further increase the chemical potential to further increase the density. Look at the pair distribution function g(r) and compare the number of particles with snapshots of the configurations (coord.gp). You may see something different from what you expect for a regular solid: do you understand what’s going on?

Documentation

************** folder contents ***************************

In this directory you find the following files:

`README.pdf`	* this file
`worm.f`	* a fortran77 implementation of the WA
`worm.h`	* a file included by worm.f at compile time
`worm.in`	* a sample input file
`worm.in_save`	* a spare copy of the input file
`statforw.f`	* a fortran77 code to calculate weighted averages and statistical errors.
	* gnuplot command files to visualize data from the output files of the simulations:
`etot.gp`	- the energy per particle, block and cumulative averages (data from '`worm.out`');
`np.gp`	- the number of particles, block and cumulative averages (data from '`worm.out`');
`rhos.gp`	- the superfluid fraction, block and cumulative averages (data from '`worm.out`');
`gofr.gp`	- the pair distribution function, block and cumulative averages (data from '`gofr`');
`nofr.gp`	- the one-body density matrix, cumulative average (data from '`nofr`');
`sofk.gp`	- the static structure factor (data from '`sofk`');
`coord.gp`	- snapshot of the current configuration of the paths (data from '`restart.coord`').

************** input description ******************************

Each row of the input file starts with a keyword (see the input file provided).
Each keyword requires a (fixed) number of arguments, as follows:

keyword `ndim`
  arg. 1: space dimensionality

keyword `type`
  arg. 1: the name of a type of particles
  arg. 2: the value of λ for this this type of particles

keyword `pbc`
  arg. 1: the side of the simulation box in direction x
  arg. 2: the side of the simulation box in direction y
  ...

keyword `mu`
  arg. 1: the name of a type of particles (as given in keyword "type")
  arg. 2: the chemical potential μ for this type

keyword `temperature`
  arg. 1: the temperature T

keyword `ntau` (max. value 1000)
  arg. 1: the number P of "time" slices; the time step is $\epsilon = 1/(PT)$

keyword `restart`
  arg. 1: simulation starts from an empty cell (0)
    or from restart files of the previous run (1).
    (Note: do not use `restart 1` if you change `ntau`)

keyword `c0`
  arg. 1: a number which controls the frequency of the
    Z sector vs the G sector (increase/decrease
    this number to favor/hinder the G sector)

keyword `v2` (specifies the pair potential between type-i and type-j particles)
  arg. 1: name of type-i particles
  arg. 2: name of type-j particles
  arg. 3: name of an output file with the tabulated potential
  arg. 4: name of the subroutine which calculates this pair potential

keyword `m_bar` (size of the move; max. value = min{200, `ntau`})
  arg. 1: maximum number of time slices involved in a move

keyword `verme` (this <u>must</u> be the last keyword and specifies the run)
  arg. 1: number of blocks
  arg. 2: number of “equilibration” blocks discarded from cumulative averages
  arg. 3: number of calculation of physical properties in each block
  arg. 4: number of moves between each calculation of physical properties

************ changing parameters on the fly ********************

Some keywords can be changed during the run by putting the corresponding keywords-values in a file called '`reset`', e.g. the command

```
your_prompt$ echo temperature 1.0 > reset
```

will reset the temperature to 1 starting from the next block.

You can change the values of `mu, temperature and c0`. When a parameter is changed, the cumulative averages are reset.
You can also use the '`reset`' file to reset the cumulative averages without changing any parameter,

```
your_prompt$ echo reset > reset
```

or to stop the run,

```
your_prompt$ echo stop > reset
```

************** output description *****************************

At the end of each block the code prints
on the standard output something like this:

```
 ===>> block            9
advance         10374. 0.400
recede          10429. 0.405
insert           2046. 0.107
remove            315. 0.717
open             2017. 0.133
close             341. 0.768
wiggle           1961. 0.617
swap             9758. 0.370
displace            0. 0.000
swtype              0. 0.000
  0.14000000000E+00  0.16250000000E+00 0.13E-01   0.10000000000E+03 zsector
  0.49200740299E+01  0.54712978054E+01 0.34E+00   0.14000000000E+02 etot
  0.36093597442E+01  0.40228954385E+01 0.38E+00   0.14000000000E+02 ekin
  0.13107142857E+01  0.14484023669E+01 0.81E-01   0.14000000000E+02 epot
  0.25285714286E+02  0.25553846154E+02 0.24E+00   0.14000000000E+02 np
  0.43256717622E+00  0.43715415540E+00 0.41E-02   0.14000000000E+02 rho
  0.76351393780E+00  0.82201711356E+00 0.26E+00   0.14000000000E+02 rhos_sd
```

(i) the block index
(ii) the number of attempted moves and the acceptance rate
  for each type of move.
(iii) for various quantities (specified by the string in the last column):
  first column: block average
  second column: cumulative average
  third column: estimated statistical error (assuming independent blocks)
  fourth column: statistical weight of this block average

Furthermore there are some output files:

`v_sd_sd`	tabulated potential for the energy (filename specified in input)
`gofr`	pair distribution function $g(r)$ (distance, block average, cumulative average, estimated statistical error, statistical weight).
nofr	one-body density matrix $n(r)$ (distance, cumulative average)
sofk	static structure factor $S(k)$ (k_x, k_y , cumulative average)
restart.coord	a file which contains coordinates for restart
restart.ind	a file which contains indices for restart

******************* Miscellaneous ***************************

* To compile a fortran code (e.g. `worm.f`):

```
your_prompt$ gfortran -O3 worm.f -o worm.x
```

(without the `-o` option the name of the executable is `'a.out'`; without the `-O3` option the code will run much slower).
On some versions of gfortran you may need to add the option `-fallow-argument-mismatch`.

* To use one of the provided gnuplot command files, say `'np.gp'`:

```
your_prompt$ gnuplot
gnuplot> l 'np.gp'
```

exit with `^C` (Control-C), then quit gnuplot.

* Various ways to redirect the standard output of the executable '`worm.x'` to a file '`worm.out`':

```
your_prompt$ ./worm.x > worm.out
```

(if '`worm.out`' already exists, it is overwritten);

```
your_prompt$ ./worm.x >> worm.out
```

(appends to '`worm.out`');

```
your_prompt$ ./worm.x | tee worm.out
```

sends a copy of the output on the screen and a copy on '`worm.out`';
if this file already exists it is overwritten;

```
your_prompt$ ./worm.x | tee -a worm.out
```

(appends to 'worm.out').

* Why and how to use statforw.f:

The code in statforw.f calculates WEIGHTED averages; it reads TWO columns (i.e. data AND weights). In the Worm Algorithm we need weighted averages because for each block any (local or semilocal) physical property must be weighted by the number of configurations of the Z sector visited in that block.

Compile the code and optionally choose a name for the executable (here statforw.x):

```
your_prompt$ gfortran statforw.f -o statforw.x
```

Example: We assume we have redirected the standard output of the simulation to the file 'worm.out' and we want to calculate mean and statistical error of the energy (the statistical errors given in the output file assume uncorrelated blocks, which may not be the case):

```
your_prompt$ grep etot worm.out | awk '{print $1, $4}' | ./statforw.x
```

In the above line, the command "grep" selects the lines of file worm.out containing the string "etot", the command "awk" prints the first and fourth columns (block averages and block weights) of the selected lines, and the executable "./statforw.x" performs the statistical analysis of the given set of data/weigths. In the output (see example below), average is the estimated mean, sigma is the statistical error, and t corr the autocorrelation time.

```
average                                5.9452843073
n,neff              100.000000            96.894287
variance                              0.3266427287
t corr                                1.1574434965
n eff                                83.7140534896
sigma                                 0.0624650816
```