Institute for Theoretical Physics
University of Cologne

Prof. Dr. Simon Trebst
Peter Bröcker, Johannes Helmes

Computational Many-Body Physics

Assignment 3

Summer Term 2015

website: http://www.thp.uni-koeln.de/trebst/Lectures/2015-CompManyBody.shtml
due date: Monday, June 1st, 18:00 - send solutions to gerlach [at] thp.uni-koeln.de

7. Nobody is perfect

programming techniques

On the previous sheet, you learned how to perform a binning analysis in order to obtain an error estimate on your Monte Carlo data. These error bars should of course also be included in your plots and learning how to do this in python is the focus of this exercise. You will also learn how to conveniently save data for later processing.

The simplest way to store data is to use the **csv format**. Its advantage over binary formats (that would take up less space) is that it is human-readable. We will use the functions savetxt and loadtxt provided by the `numpy` module. Let us assume that we want to save data for a plot with x- and y values as well as errors on the y values. Each of these sets of values is represented as an array. The first step towards saving it is to merge them into a single array, using the function vstack and to then save them in a second step:

```
import numpy.random as npr

# generate random data
x = npr.random(10)
y = npr.random(10)
yerr = npr.random(10)

# join data vertically -v-stack to obtain a data matrix
# also try hstack and see what happens!
data = np.vstack((x, y, yerr)).T

# then save data
np.savetxt('data.csv', data, delimiter=',')
```

If you want to load them back into python for plotting, all we have to do is use `loadtxt`:

```
data = loadtxt('data.csv', delimiter=',')
```

The data is now present in the form of a matrix and we can use the usual slice operations to access the respective columns. Try it out by printint the matrix and studying its structure!

Just in case you were wondering what the keyword `delimiter` does: We have to choose a special character that separates two values. Originally, csv stood for *comma separated values* but is now also often refered to *character separated values* because of the freedom to choose the precise delimiting character.

In a final step, we would like to display the data including error bars. We can the `matplotlib`

module and its function `errorbar` and supply it with the x and y values as well as the errors which we pass using the keyword `yerr`:

```
plt.errorbar(x, y, yerr=yerr, fmt='o')
```

If your x values also carry an uncertainty, you can also pass another array using `xerr` to account for that.

8. Extended Ensemble Simulations *10 points + 10 bonus points*

In this third exercise we will employ extended ensemble simulation techniques to (re)investigate the 2D Ising model, now performing an "all-temperature" calculation allowing us to directly estimate the density of states $g(E)$. In the second part we will consider a prototypical system undergoing a first-order phase transition, the Q-state Potts model.

1. Implement the **Wang-Landau algorithm** for the 2D Ising model on the square lattice with periodic boundary conditions. (You should be able to simply expand your single spin-flip Metropolis code.)
2. Calculate estimates for the **density of states** $g(E)$ from Wang-Landau sampling for systems of linear size $L = 8, 16, 32$ in the energy range $E_{\min} = -2N$ to $E_{\max} = 0$. Normalize the calculated density of states such that $g(E_{\min}) = 2$ and plot $\ln g(E)$ for the different system sizes.
3. From the calculated estimate of the density of states calculate the following thermodynamic observables and plot them in the temperature range $T \in [0,4]$ (showing data for all three system sizes in one plot):
   - the **energy** $U(T) = \frac{1}{Z}\sum_E g(E)E\exp(-\beta E) = \langle E\rangle_T$
   - the **specific heat** $C_v(T) = dU/dT = \left(\langle E^2\rangle_T - \langle E\rangle_T^2\right)/T^2$
   - the **free energy** $F(T) = -T\log Z$
   - the **entropy** $S(T) = \left(U(T) - F(T)\right)/T$

   where Z is the partition function $Z = \sum_E g(E)\cdot\exp(-\beta E)$.
   Compare these results to those obtained in the last exercise.
4. Expand the above implementation of the Wang-Landau algorithm to simulate the **Potts model**, which for a Q-state Potts spin $\sigma_i \in \{1, 2, \ldots, Q\}$ we define via the Hamiltonian

$$H = -\sum_{\langle ij\rangle} \delta\left(\sigma_i, \sigma_j\right),$$

   where the sum again runs over all nearest-neighbor bonds. Consider a periodic square lattice with $N = L^2$ Potts spins and estimate the density of states $g(E)$ for the 10-state Potts model ($Q = 10$) in the energy range $E_{\min} = -2N$ to $E_{\max} = 0$. This estimate can now be normalized such that $g(E_{\min}) = Q = 10$.
5. For systems of linear size $L = 8, 16, 32$ again plot **thermodynamic averages** for the energy, specific heat, free energy, and entropy as above.
6. Plot the ***canonical*** **distribution function** $P(E) = g(E)\exp(-\beta E)$ in proximity of the

thermal phase transition, which occurs at temperature

$$T^* = \frac{1}{\ln(1+\sqrt{Q})}$$

for the *infinite* system ($L = \infty$). Can you observe a double-peak structure indicative of a first-order phase transition?

7. *Optional exercise I:* From the precise location of the double-peak structure in the canonical distribution function $P(E)$ you can determine estimates of the **finite-size transition temperature** $T^*(L)$ for a given linear system size L. Plot your estimates versus the inverse system size $1/L^2$ and extrapolate your data to the infinite system size limit – can you recover the analytical estimate above? You might want to calculate a few extra system sizes $L = 8, 16, 24, 32, 48, 64, \ldots$ for this extrapolation.

8. *Optional exercise II:* Calculate the **local diffusivity** $D(E)$ of the random walk in energy space for the Ising and Potts model simulations above by running simulations with *fixed* weights $w(E) \approx 1/g(E)$, which you have obtained from Wang-Landau sampling. To obtain an estimate for the local diffusivity

   - Record two histograms during the sampling process – the energy histogram $h(E)$, which is incremented for every step, and the histogram $h^+(E)$, which is incremented only if the last extremal energy you have visited is $E_{\min}$ (and *not* $E_{\max}$).
   - From these two histograms calculate the fraction $f(E) = h^+(E)/h(E)$, which estimates how much time on average the energy random walker spends at a given energy E diffusing towards higher energies. Plot this fraction $f(E)$.
   - Calculate the derivative df/dE of this fraction (and plot it).
   - Plot an estimate of the local diffusivity via

$$D(E) \propto \left(h(E) \cdot \frac{df}{dE} \right)^{-1} .$$

To get started, we have again prepared a skeleton program. All you have to do is implement the update and measurement routines. Note that the way the lattice is set up, the code is geared towards the Ising model but the algorithmic structure of the Potts model is really so similar that you can use the same skeleton with just minor modifications.