Institute for Theoretical Physics
University of Cologne

Prof. Dr. Simon Trebst
Peter Bröcker, Johannes Helmes

Computational Many-Body Physics

Assignment 6

Summer Term 2015

website: http://www.thp.uni-koeln.de/trebst/Lectures/2015-CompManyBody.shtml
due date: Monday, July 13th, 18:00 - send solutions to helmes [at] thp.uni-koeln.de

14. Certainly uncertain

Programming technique

When using numerical methods based on the statistical evaluation of observables we inevitably have to deal with expectation values coming with statistical uncertainties. Luckily, Python provides a nice package, which allows to handle precisely this situation, i.e. store numbers and their error estimates of the form $r = 0.372 \pm 0.003$. The package can even perform error propagation when applying arithmetics to these numbers. In the following, we introduce the basic features of this package. You can install the package `uncertainties` e.g. using the pip command

```
pip install uncertainties
```

Entering values with their corresponding errors can be done in various ways (see the example below) and printing works as usual:

```python
import uncertainties as unc

x=unc.ufloat_fromstr("0.23+/-0.03")
y=unc.ufloat_fromstr("0.0042(2)")
z=unc.ufloat(1.223,0.007)

print z, x.nominal_value, y.std_dev
```

An arithmetic operation can be applied to these numbers using the common syntax. However, built-in mathematical functions must be imported from the `uncertainties.umath` subpackage.

```python
from uncertainties.umath import sqrt, cos, sin # ...

a=x+y
b=z**2
c=sqrt(x*5 + z)
```

Especially useful are uncertainties' operations on arrays, which can be accessed in two different ways. The second approach has the advantage that it can use all of the powerful numpy array operations.

```
#first alternative
from uncertainties import unumpy

values=unumpy.uarray([2.5,2.3,2.8],[0.03,0.02,0.05])
cosines=unumpy.cos(values)

#second alternative
import numpy as np
import uncertainties as unc

values=np.array([unc.ufloat(2.5,0.03),unc.ufloat(2.3,0.02),unc.ufloat(2.8,0.05)])
total=np.sum(values)
mean=np.mean(values)

print mean
```

15. Stochastic Series Expansion of the Heisenberg model

10+2 points

In this exercise we want to find footprints of a thermal phase transition in the three-dimensional spin-$\frac{1}{2}$ Heisenberg model on the cubic lattice, which is described by the Hamiltonian

$$H = -J \sum_{\langle i,j \rangle} \vec{S}_i \vec{S}_j \,. \tag{1}$$

For simplicity, we set the ferromagnetic coupling constant to $J = 1$ and sum runs over all pairs of nearest neighbors of the cubic lattice.

We want to apply the quantum Monte Carlo approach based on a stochastic series expansion (SSE) to investigate this model. SSE is based on rewriting the Boltzmann factor in the partition function as a Taylor expansion

$$\begin{aligned} Z &= \mathrm{Tr} e^{-\beta H} & (2)\\ &= \sum_{\alpha} \sum_{n=0}^{\infty} \left\langle \alpha \left| \frac{(-\beta H)^n}{n!} \right| \alpha \right\rangle . & (3) \end{aligned}$$

It is particularly useful to rewrite the Hamiltonian in terms of contributions of every bond

$$\begin{aligned} H &= -\sum_b H_b & (4)\\ &= -\sum_b \left[\underbrace{\left(\vec{S}^z_{b_1} \vec{S}^z_{b_2} - \frac{1}{4} \right)}_{\text{diagonal}} + \underbrace{\frac{1}{2} \left(\vec{S}^+_{b_1} \vec{S}^-_{b_2} + \vec{S}^-_{b_1} \vec{S}^+_{b_2} \right)}_{\text{off-diagonal}} \right] + const. & (5) \end{aligned}$$

The constant absorbs the shift of the Hamiltonian induced by the artificially introduced addition of $\frac{1}{4}$ per bond. This form of the Hamiltonian is perfectly adapted to the needs of SSE, namely that all non-zero matrix elements ($\langle \uparrow\uparrow | H_b | \uparrow\uparrow \rangle, \langle \downarrow\downarrow | H_b | \downarrow\downarrow \rangle, \langle \uparrow\downarrow | H_b | \downarrow\uparrow \rangle, \langle \downarrow\uparrow | H_b | \uparrow\downarrow \rangle$) have the same value of $\frac{1}{2}$ in this case.

1. Determine the probability for the insertion and removal of a bond operator into the SSE configuration in the *diagonal* update part.
2. Figure out how the *off-diagonal* (loop) update has to look like. Loops are deterministic and you should identify every possible loop and flip it with probability $\frac{1}{2}$. Do not forget single spins which are not connected to any bond operators.
3. Implement an SSE algorithm for the three-dimensional Heisenberg model on a square lattice. In general, we would have to set a maximal expansion order M which is adapted during the thermalization phase, but for this exercise you can fix it to $M = 2\beta L^3$.
4. Perform simulations for $\beta = 0.2, 0.4, 0.6, \ldots 5.0$ and $L = 8, 12, 16$. Measure the absolute **magnetization** per spin and plot it against the inverse temperature β. It should be sufficient to do 10^5 to 10^6 measurement steps (sweeps) after a considerable thermalization time (usually 10% of the number of sweeps).
5. Remember that the **energy** can be elegantly obtained by simply measuring the expansion order n as $\langle E \rangle = -\frac{\langle n \rangle}{\beta}$. Plot the energy per spin against the inverse temperature β. Can you identify signatures of a phase transition?

*6. The SSE method exploits the fact that only very few expansion orders significantly contribute to the partition sum (3). Can you explain why? Check your hypothesis by generating a histogram that displays the relative frequency of the expansion orders n in the Monte Carlo Markov chain. How do things change when you vary the inverse temperature β?