Project 4, deadline November 14

Computational Physics I FYS3150/FYS4150

Department of Physics, University of Oslo, Norway

Fall semester 2016

Studies of phase transitions in magnetic systems

Introduction. The aim of this project is to study a widely popular model to simulate phase transitions, the so-called Ising model in two dimensions. At a given critical temperature, this model exhbits a phase transition from a magnetic phase (a system with a finite magnetic moment) to a phase with zero magnetization.

In its simplest form the energy of the Ising model is expressed as, without an externally applied magnetic field,

$$E = -J \sum_{<kl>}^{N} s_k s_l$$

with $s_k = \pm 1$. The quantity N represents the total number of spins and J is a coupling constant expressing the strength of the interaction between neighboring spins. The symbol $< kl >$ indicates that we sum over nearest neighbors only. We will assume that we have a ferromagnetic ordering, viz $J > 0$. We will use periodic boundary conditions and the Metropolis algorithm only. The material on the Ising model can be found in chapter 13 of the lecture notes. The Metropolis algorithm is discussed in chapter 12.

For this project you can collaborate with fellow students but you have to hand in an individual report. This project (together with projects 3 and 5) counts 1/3 of the final mark.

Project 4a): A simple 2×2 lattice, analytical expressions. Assume we have only two spins in each dimension, that is $L = 2$. Find the analytical expression for the partition function and the corresponding expectations values for the energy E, the mean absolute value of the magnetic moment $|M|$ (we will refer to this as the mean magnetization), the specific heat C_V and the susceptibility χ as functions of T using periodic boundary conditions.

Project 4b): Writing a code for the Ising model. Write now a code for the Ising model which computes the mean energy E, mean magnetization $|M|$, the specific heat C_V and the susceptibility χ as functions of T using periodic boundary conditions for $L = 2$ in the x and y directions. Compare your results with the expressions from a) for a temperature $T = 1.0$ (in units of kT/J).

How many Monte Carlo cycles do you need in order to achieve a good agreeement?

Project 4c): When is the most likely state reached? We choose now a square lattice with $L = 20$ spins in the x and y directions.

In the previous exercise we did not study carefully how many Monte Carlo cycles were needed in order to reach the most likely state. Here we want to perform a study of the time (here it corresponds to the number of Monte Carlo sweeps of the lattice) one needs before one reaches an equilibrium situation and can start computing various expectations values. Our first attempt is a rough and plain graphical one, where we plot various expectations values as functions of the number of Monte Carlo cycles.

Choose first a temperature of $T = 1.0$ (in units of kT/J) and study the mean energy and magnetisation (absolute value) as functions of the number of Monte Carlo cycles. Let the number of Monte Carlo cycles (sweeps per lattice) represent time. Use both an ordered (all spins pointing in one direction) and a random spin orientation as starting configuration. How many Monte Carlo cycles do you need before you reach an equilibrium situation? Repeat this analysis for $T = 2.4$. Can you, based on these values estimate an equilibration time? Make also a plot of the total number of accepted configurations as function of the total number of Monte Carlo cycles. How does the number of accepted configurations behave as function of temperature T?

Project 4d): Analyzing the probability distribution. Compute thereafter the probability $P(E)$ for the previous system with $L = 20$ and the same temperatures, that is at $T = 1.0$ and $T = 2.4$. You compute this probability by simply counting the number of times a given energy appears in your computation. Start the computation after the steady state situation has been reached. Compare your results with the computed variance in energy σ_E^2 and discuss the behavior you observe.

Studies of phase transitions. Near T_C we can characterize the behavior of many physical quantities by a power law behavior. As an example, for the Ising class of models, the mean magnetization is given by

$$\langle M(T) \rangle \sim (T - T_C)^{\beta},$$

where $\beta = 1/8$ is a so-called critical exponent. A similar relation applies to the heat capacity

$$C_V(T) \sim |T_C - T|^{\alpha},$$

and the susceptibility

$$\chi(T) \sim |T_C - T|^{\gamma} , \tag{1}$$

with $\alpha = 0$ and $\gamma = 7/4$. Another important quantity is the correlation length, which is expected to be of the order of the lattice spacing for $T >> T_C$. Because the spins become more and more correlated as T approaches T_C, the correlation length increases as we get closer to the critical temperature. The divergent behavior of ξ near T_C is

$$\xi(T) \sim |T_C - T|^{-\nu} . \tag{2}$$

A second-order phase transition is characterized by a correlation length which spans the whole system. Since we are always limited to a finite lattice, ξ will be proportional with the size of the lattice. Through so-called finite size scaling relations it is possible to relate the behavior at finite lattices with the results for an infinitely large lattice. The critical temperature scales then as

$$T_C(L) - T_C(L = \infty) = aL^{-1/\nu}, \tag{3}$$

with a a constant and ν defined in Eq. (2). We set $T = T_C$ and obtain a mean magnetisation

$$\langle \mathcal{M}(T) \rangle \sim (T - T_C)^{\beta} \rightarrow L^{-\beta/\nu}, \tag{4}$$

a heat capacity

$$C_V(T) \sim |T_C - T|^{-\gamma} \rightarrow L^{\alpha/\nu}, \tag{5}$$

and susceptibility

$$\chi(T) \sim |T_C - T|^{-\alpha} \rightarrow L^{\gamma/\nu}. \tag{6}$$

Project 4e): Numerical studies of phase transitions. We wish to study the behavior of the Ising model in two dimensions close to the critical temperature as a function of the lattice size $L \times L$. Calculate the expectation values for $\langle E \rangle$ and $\langle |M| \rangle$, the specific heat C_V and the susceptibility χ as functions of T for $L = 40$, $L = 60$, $L = 100$ and $L = 140$ for $T \in [2.0, 2.3]$ with a step in temperature $\Delta T = 0.05$ or smaller. You may find it convenient narrow the domain for T.

Plot $\langle E \rangle$, $\langle |M| \rangle$, C_V and χ as functions of T. Can you see an indication of a phase transition? Use the absolute value $\langle |M| \rangle$ when you evaluate χ. For these production runs you should parallelize the code using MPI (recommended). Alternatively OpenMP can be used. Use optimization flags when compiling. Perform a timing analysis of some selected runs in order to see that you get an optimal speedup when parallelizing your code.

Project 4f): Extracting the critical temperature. Use Eq. (3) and the exact result $\nu = 1$ in order to estimate T_C in the thermodynamic limit $L \to \infty$ using your simulations with $L = 40$, $L = 60$, $L = 100$ and $L = 140$ The exact result for the critical temperature (after Lars Onsager) is $kT_C/J = 2/ln(1 + \sqrt{2}) \approx 2.269$ with $\nu = 1$.

Background literature

If you wish to read more about the Ising model and statistical physics here are three suggestions.

- M. Plischke and B. Bergersen, *Equilibrium Statistical Physics*, World Scientific, see chapters 5 and 6.
- D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, Cambridge, see chapters 2,3 and 4.
- M. E. J. Newman and T. Barkema, *Monte Carlo Methods in Statistical Physics*, Oxford, see chapters 3 and 4.

Introduction to numerical projects

Here follows a brief recipe and recommendation on how to write a report for each project.

- Give a short description of the nature of the problem and the eventual numerical methods you have used.
- Describe the algorithm you have used and/or developed. Here you may find it convenient to use pseudocoding. In many cases you can describe the algorithm in the program itself.
- Include the source code of your program. Comment your program properly.
- If possible, try to find analytic solutions, or known limits in order to test your program when developing the code.
- Include your results either in figure form or in a table. Remember to label your results. All tables and figures should have relevant captions and labels on the axes.
- Try to evaluate the reliabilty and numerical stability/precision of your results. If possible, include a qualitative and/or quantitative discussion of the numerical stability, eventual loss of precision etc.
- Try to give an interpretation of you results in your answers to the problems.

- Critique: if possible include your comments and reflections about the exercise, whether you felt you learnt something, ideas for improvements and other thoughts you've made when solving the exercise. We wish to keep this course at the interactive level and your comments can help us improve it.
- Try to establish a practice where you log your work at the computerlab. You may find such a logbook very handy at later stages in your work, especially when you don't properly remember what a previous test version of your program did. Here you could also record the time spent on solving the exercise, various algorithms you may have tested or other topics which you feel worthy of mentioning.

Format for electronic delivery of report and programs

The preferred format for the report is a PDF file. You can also use DOC or postscript formats or as an ipython notebook file. As programming language we prefer that you choose between C/C++, Fortran2008 or Python. The following prescription should be followed when preparing the report:

- Use Devilry to hand in your projects, log in at `http://devilry.ifi.uio.no` with your normal UiO username and password and choose either 'fys3150' or 'fys4150'. There you can load up the files within the deadline.
- Upload **only** the report file! For the source code file(s) you have developed please provide us with your link to your github domain. The report file should include all of your discussions and a list of the codes you have developed. Do not include library files which are available at the course homepage, unless you have made specific changes to them.
- In your git repository, please include a folder which contains selected results. These can be in the form of output from your code for a selected set of runs and input parametxers.
- In this and all later projects, you should include tests (for example unit tests) of your code(s).
- Comments from us on your projects, approval or not, corrections to be made etc can be found under your Devilry domain and are only visible to you and the teachers of the course.

Finally, we encourage you to work two and two together. Optimal working groups consist of 2-3 students. For this specific report you need to hand in an individual report.

How to install openmpi and/or OpenMP on your PC/laptop

If you use your own laptop, for linux/ubuntu users, you need to install two packages (alternatively use the synaptic package manager)

```
  sudo apt-get install libopenmpi-dev
  sudo apt-get install openmpi-bin
```

For OS X users, install brew (after having installed xcode and gcc, needed for the gfortran compiler of openmpi) and then run

```
brew install open-mpi
```

When compiling from the command line, depending on your choice of programming language you need to compile and link as for example

```
mpic++ -O3 -o <executable> <programname.cpp>
```

if you use c++ (alternatively mpicxx) and

```
mpif90 -O3 -o <executable> <programname.f90>
```

if you use Fortran90.

When running an executable, run as

```
mpirun -n 10 ./<executable>
```

where -n indicates the number of processes, 10 here.

With openmpi installed, when using Qt, add to your .pro file the instructions at Svenn-Arne Dragly's site

You may need to tell Qt where opempi is stored.

For the machines at the computer lab, openmpi is located at /usr/lib64/openmpi/bin Add to your .bashrc file the following

```
export PATH=/usr/lib64/openmpi/bin:$PATH
```

For Windows users we recommend to follow the instructions at the Open MPI site.

If you use OpenMP, for linux users, compile and link with for example

```
c++ -O3 -fopenmp -o <executable>  <programname.cpp>
```

For OS X users, you need to install clang-omp using brew, that is

```
brew install clang-omp
```

and then compile and link with for example

```
clang-omp++ -O3 -fopenmp -o <executable>  <programname.cpp>
```

If you program in Fortran and use **gfortran**, compile as for example

```
gfortran -O3 -fopenmp -o <executable>  <programname.f90>
```

If you have access to Intel's **ifort** compiler, compile as

```
ifort -O3 -fopenmp -o <executable>  <programname.f90>
```