Computational Physics - Exercise 3

Joel Schumacher, 309231, joel.schumacher@rwth-aachen.de

May 31, 2017

1D Ising model

Introduction

The Ising model is a simple and popular theoretical model for solids that exhibits a magnetic phase transition. Without an external magnetic field our Hamiltonian is as follows:

$$H = -J \sum_{\text{neighbours i,j}} S_i \cdot S_j$$

where S_i is the a spin operator with two possible eigenvalues. In the case of this simulation the constants are chosen so that $J = 1$ and S_i has two possible eigenvalues $+1$ and -1. Also k_B is set to $k_B = 1$.

In the specific case of the 1D Ising model the total energy is then:

$$E = -\sum_{n=1}^{N-1} S_n S_{n+1}$$

assuming free boundary conditions and N spins in our system.

Simulation model

For our simulation the Metropolis Monte Carlo (MMC) method is used. The system of spins is initialized with a random array of N values in $\{-1, 1\}$ representing the spins of the system. Then $N_{wait} = N_{samples}/10$ steps of relaxation for the system are performed after which $N_{samples}$ regular steps of measurement follow. In each of these steps N substeps are performed, which consist of the selection of a (uniform) random spin in the system and the calculation of the total change of energy ΔE of the system if that spin was flipped. If $\Delta E < 0$, the flip is applied, if $\Delta E > 0$ the flip is only applied if $\exp(-\Delta E \cdot \beta) > r$, where r is a uniform random variable in $[0, 1]$. This represents a Markov Chain that generates a sequence of states Ω of the system, that follows the probability distribution of the grandcanonical ensemble. Using this sequence the expectation value of E and E^2 and in turn $U = \langle E \rangle$ and $C = \beta^2 \left(\langle E^2 \rangle - \langle E \rangle^2 \right)$ can be calculated through the arithmetic mean:

$$< f > \approx \frac{1}{\#\Omega} \sum_{\{S_1,\ldots,S_N\}} f(S_1, \ldots, S_N).$$

Simulation results

The results of the simulation, measuring the average energy U/N and specific heat C/N per spin and comparing them to the theoretical predictions,

$$U_{theory} = -\frac{N-1}{N} \tanh \beta$$

$$C_{theory} = \frac{N-1}{N} \left(\beta / \cosh \beta\right)^2$$

can be seen in table 1 and 2 with $N_{samples} = 1000$ and $N_{samples} = 10000$ respectively. σ_{rel} denote relative errors, e.g. $\sigma_{U,rel} = \left| \frac{U/N - U_{theory}}{U_{theory}} \right|$. For $N_{samples} = 1000$ the relative errors for the energy is always smaller than 10^{-2} and the relative error the heat capactiy is almost always smaller than 10^{-2}, but increases rapidly when approaching

T	β	U/N	C/N	U_{theory}	C_{theory}	$\sigma_{U,rel}$	$\sigma_{C,rel}$
4.000E+00	2.500E-01	-2.449E-01	5.391E-02	-2.447E-01	5.869E-02	1.056E-03	8.156E-02
3.800E+00	2.632E-01	-2.574E-01	6.539E-02	-2.570E-01	6.460E-02	1.574E-03	1.219E-02
3.600E+00	2.778E-01	-2.700E-01	7.042E-02	-2.706E-01	7.143E-02	2.063E-03	1.417E-02
3.400E+00	2.941E-01	-2.849E-01	7.562E-02	-2.856E-01	7.935E-02	2.459E-03	4.704E-02
3.200E+00	3.125E-01	-3.032E-01	8.493E-02	-3.024E-01	8.862E-02	2.477E-03	4.158E-02
3.000E+00	3.333E-01	-3.212E-01	9.718E-02	-3.212E-01	9.953E-02	1.394E-04	2.355E-02
2.800E+00	3.571E-01	-3.426E-01	1.128E-01	-3.424E-01	1.125E-01	7.764E-04	2.782E-03
2.600E+00	3.846E-01	-3.641E-01	1.260E-01	-3.663E-01	1.279E-01	6.115E-03	1.509E-02
2.400E+00	4.167E-01	-3.933E-01	1.411E-01	-3.937E-01	1.465E-01	1.068E-03	3.717E-02
2.200E+00	4.545E-01	-4.250E-01	1.723E-01	-4.252E-01	1.690E-01	5.328E-04	1.968E-02
2.000E+00	5.000E-01	-4.623E-01	2.052E-01	-4.617E-01	1.964E-01	1.332E-03	4.473E-02
1.800E+00	5.556E-01	-5.027E-01	2.355E-01	-5.042E-01	2.298E-01	2.987E-03	2.475E-02
1.600E+00	6.250E-01	-5.540E-01	2.819E-01	-5.540E-01	2.702E-01	3.451E-05	4.317E-02
1.400E+00	7.143E-01	-6.110E-01	3.207E-01	-6.127E-01	3.179E-01	2.800E-03	8.778E-03
1.200E+00	8.333E-01	-6.822E-01	3.330E-01	-6.816E-01	3.708E-01	9.426E-04	1.020E-01
1.000E+00	1.000E+00	-7.620E-01	3.801E-01	-7.608E-01	4.196E-01	1.484E-03	9.409E-02
8.000E-01	1.250E+00	-8.492E-01	4.211E-01	-8.474E-01	4.377E-01	2.061E-03	3.788E-02
6.000E-01	1.667E+00	-9.295E-01	3.623E-01	-9.302E-01	3.692E-01	7.810E-04	1.874E-02
4.000E-01	2.500E+00	-9.750E-01	5.462E-01	-9.856E-01	1.660E-01	1.083E-02	2.290E+00
2.000E-01	5.000E+00	-9.799E-01	1.289E+00	-9.989E-01	4.535E-03	1.901E-02	2.831E+02

Table 1: Results of the simulation for the 1D Ising model with $N = 1000$, $N_{samples} = 1000$.

lower temperatures $0.4K$ and $0.2K$. For $N_{samples} = 10000$ the agreement is even better with a relative error for the energy that is about two orders of magnitude smaller, but the same behaviour for the heat capacity can be observed again.

Discussion

It can be seen that the simulation generally gives a good prediction for both the average energy per spin U/N and specific heat capacity per spin C/N in the one-dimensional case if the temperature is not close to zero. For low temperatures, $T = 0.2$ is the most obvious, the simulation does not give acceptable results for the heat capacity. This behaviour persists even for a higher number of samples, which suggests that the MMC method might be a generally inadequate method for this case.

2D Ising model

Introduction

In the two-dimensional model the total energy of the system becomes (with free boundary conditions):

$$E = -\sum_{i=1}^{N-1}\sum_{j=1}^{N} S_{i,j}S_{i+1,j} - \sum_{i=1}^{N}\sum_{j=1}^{N-1} S_{i,j}S_{i,j+1}$$

In contrast to the one-dimensional model further analysis is carried out regarding the magnetic properties of the system. The average magnetization per spin is:

$$M/N^2 = \left\langle \sum_{i,j=1}^{N} S_{i,j} \right\rangle = \frac{\sum_{\{S_{1,1},\ldots,S_{N,N}\}} \exp(-\beta E) \sum_{i,j=1}^{N} S_{i,j}}{\sum_{\{S_{1,1},\ldots,S_{N,N}\}} \exp(-\beta E)}$$

Simulation model

For the two-dimensional Ising model the only changes from the one-dimensional case are the calculation of the energy-difference for obvious geometric reasons and the increase from N to N^2 substeps per sample to maintain the system-scale-independence of a step of the MMC method.

T	β	U/N	C/N	U_{theory}	C_{theory}	$\sigma_{U,rel}$	$\sigma_{C,rel}$
4.000E+00	2.500E-01	-2.443E-01	5.882E-02	-2.447E-01	5.869E-02	1.361E-03	2.248E-03
3.800E+00	2.632E-01	-2.566E-01	6.560E-02	-2.570E-01	6.460E-02	1.585E-03	1.536E-02
3.600E+00	2.778E-01	-2.710E-01	7.112E-02	-2.706E-01	7.143E-02	1.678E-03	4.292E-03
3.400E+00	2.941E-01	-2.855E-01	8.012E-02	-2.856E-01	7.935E-02	3.548E-04	9.618E-03
3.200E+00	3.125E-01	-3.025E-01	8.667E-02	-3.024E-01	8.862E-02	2.314E-04	2.196E-02
3.000E+00	3.333E-01	-3.208E-01	1.005E-01	-3.212E-01	9.953E-02	1.300E-03	9.889E-03
2.800E+00	3.571E-01	-3.427E-01	1.108E-01	-3.424E-01	1.125E-01	9.014E-04	1.503E-02
2.600E+00	3.846E-01	-3.668E-01	1.261E-01	-3.663E-01	1.279E-01	1.321E-03	1.387E-02
2.400E+00	4.167E-01	-3.941E-01	1.442E-01	-3.937E-01	1.465E-01	8.573E-04	1.577E-02
2.200E+00	4.545E-01	-4.250E-01	1.691E-01	-4.252E-01	1.690E-01	4.585E-04	2.589E-04
2.000E+00	5.000E-01	-4.614E-01	1.935E-01	-4.617E-01	1.964E-01	4.797E-04	1.504E-02
1.800E+00	5.556E-01	-5.039E-01	2.278E-01	-5.042E-01	2.298E-01	5.140E-04	8.753E-03
1.600E+00	6.250E-01	-5.541E-01	2.670E-01	-5.540E-01	2.702E-01	8.461E-05	1.199E-02
1.400E+00	7.143E-01	-6.125E-01	3.281E-01	-6.127E-01	3.179E-01	4.078E-04	3.188E-02
1.200E+00	8.333E-01	-6.805E-01	3.759E-01	-6.816E-01	3.708E-01	1.644E-03	1.377E-02
1.000E+00	1.000E+00	-7.603E-01	4.146E-01	-7.608E-01	4.196E-01	7.302E-04	1.184E-02
8.000E-01	1.250E+00	-8.478E-01	4.321E-01	-8.474E-01	4.377E-01	4.629E-04	1.273E-02
6.000E-01	1.667E+00	-9.307E-01	3.556E-01	-9.302E-01	3.692E-01	5.179E-04	3.674E-02
4.000E-01	2.500E+00	-9.853E-01	1.533E-01	-9.856E-01	1.660E-01	3.481E-04	7.675E-02
2.000E-01	5.000E+00	-9.910E-01	1.995E-01	-9.989E-01	4.535E-03	7.944E-03	4.299E+01

Table 2: Results of the simulation for the 1D Ising model with $N = 1000$, $N_{samples} = 10000$.

Simulation results

The results of the simulation can be seen in figure 1. The average energy per spin rises monotonously for rising T and while the curve for $N = 10$ exhibits some noise, $N = 50$ and $N = 100$ look comparable and sufficiently smooth. The heat capacity decreases with higher temperature, while showing some noise for temperatures below the critical temperature. The average magnetization per spin clearly shows for every number of N that below the critical temperature $T_C = \frac{2}{\ln(1+\sqrt{2})}$ there is some ferromagnetic behaviour with finite and occasionally very high magnetizations. At the critical temperature universally a phase transition can be observed in that the magneziation rapidly drops to zero.

Discussion

The MMC method is not just applicable to determine the energy or the heat capacity for a 1D Ising model system, but also clearly reproduces the same magnetic phase transition exhibited by the 2D Ising model. It is easy to implement and computationally very inexpensive, making the execution very feasible on a personal computer. Combined with it's potency to analyze systems such as the Ising model it has understandably become a very popular method.

Appendix

Program for 1D Ising model

```
1 import math
2 import random
3
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 PERIODIC_BOUNDARIES = False
8
```

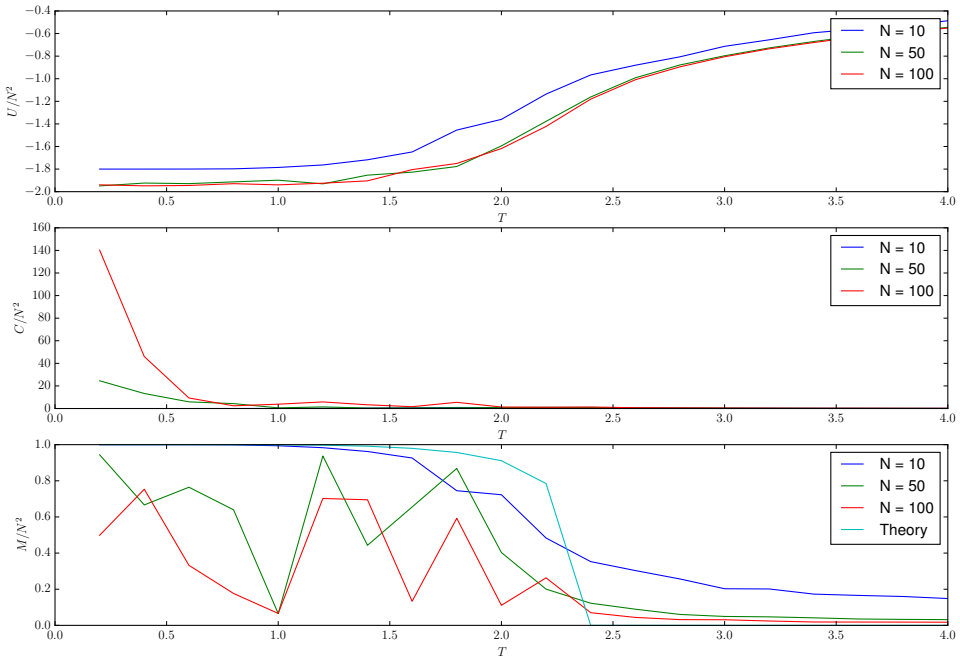

Figure 1: U/N^2, C/N^2 and M/N^2 over T for a different number of spins N.

```
#N = 10
#N = 100
N = 1000

N_SAMPLES = 1000
N_SAMPLES = 10000

def energy(spins):
    # shift elements
    shifted = np.roll(spins, -1)
    if not PERIODIC_BOUNDARIES:
        shifted[-1] = 0
    return -np.dot(spins, shifted)

def U_theory(N, beta):
    return -(N - 1)/N * np.tanh(beta)

def C_theory(N, beta):
    x = beta / np.cosh(beta)
    return (N - 1)/N * x*x

def mmc(beta):
    U = 0
    C = 0
    spins = np.random.randint(2, size=N) * 2 - 1 # N random numbers, either -1
        or 1
    e = energy(spins)
    N_WAIT = int(N_SAMPLES / 10)
    for sample in range(N_SAMPLES + N_WAIT):
```

```
37          for i in range(N):
38              j = random.randint(0, N-1)
39
40              #e_old = rest + spins[j] * (spins[j-1] + spins[j+1])
41              # e_new = rest - spins[j] * (spins[j-1] + spins[j+1])
42              # delta_e = e_new - e_old = 2 * spins[j] * (spins[j-1] + spins[j
                    +1])
43
44              # assuming spins[j] is flipped
45              delta_e = 0
46              if j > 0: delta_e += spins[j-1]
47              if j < N-1: delta_e += spins[j+1]
48              delta_e *= 2.0 * spins[j]
49
50              q = math.exp(-delta_e * beta)
51              if q > random.random():
52                  # flip the spin
53                  spins[j] = -spins[j]
54                  e += delta_e
55
56          if sample >= N_WAIT:
57              U += e
58              C += e*e
59      U = U / N_SAMPLES
60      C = beta*beta * (C / N_SAMPLES - U*U)
61
62      return U, C
63
64  print("T           beta          U/N           C/N          U_t           C_t
              err_U       err_C")
65  for T in np.arange(4.0, 0.19, -0.2):
66      k_B = 1
67      beta = 1 / (k_B * T)
68
69      U, C = mmc(beta)
70
71      U_t = U_theory(N, beta)
72      C_t = C_theory(N, beta)
73
74      print("{0:.3E}   {1:.3E}    {2:.3E}   {3:.3E}   {4:.3E}   {5:.3E}   {6:.3E}
                {7:.3E}".format(
75          T, beta, U/N, C/N, U_t, C_t, np.abs((U_t - U/N)/U_t), np.abs((C_t - C/N
                )/C_t)))
```

Program for 2D Ising model

```
1  import math
2  import random
3  from multiprocessing import Pool
4  from functools import partial
5  import json
6
7  import numpy as np
8
9  PERIODIC_BOUNDARIES = False
10
11 N_SAMPLES = 1000
```

```
12 #N_SAMPLES = 10000
13
14 def energy(spins):
15     # shift elements on y axis <=> spins[i][j] gets the value of spins[i+1][j]
16     shiftedy = np.roll(spins, -1, 0)
17     # spins[i][j] gets the value of spins[i][j+1]
18     shiftedx = np.roll(spins, -1, 1)
19
20     if not PERIODIC_BOUNDARIES:
21         shiftedy[-1,:] = 0
22         shiftedx[:,-1] = 0
23
24     # multiply matrices component wise
25     return - np.sum(np.multiply(spins, shiftedy)) - np.sum(np.multiply(spins,
           shiftedx))
26
27 def mmc(T, N):
28     print("T =", T)
29     k_B = 1
30     beta = 1 / (k_B * T)
31
32     U, C, M = 0, 0, 0
33
34     # N random numbers, either -1 or 1
35     spins = np.random.randint(2, size=(N,N)) * 2 - 1
36     e = energy(spins)
37     N_WAIT = int(N_SAMPLES / 10)
38     N_MEASUREMENTS = 0
39     for sample in range(N_SAMPLES + N_WAIT):
40         for n in range(N*N):
41             j1, j2 = random.randint(0, N-1), random.randint(0, N-1)
42
43             # assuming spins[j1,j2] is flipped
44             delta_e = 0
45             if j1 > 0: delta_e += spins[j1-1,j2]
46             if j2 > 0: delta_e += spins[j1,j2-1]
47             if j1 < N-1: delta_e += spins[j1+1,j2]
48             if j2 < N-1: delta_e += spins[j1,j2+1]
49             delta_e *= 2.0 * spins[j1,j2]
50
51             q = math.exp(-delta_e * beta)
52             if q > random.random():
53                 # flip the spin
54                 spins[j1,j2] = -spins[j1,j2]
55                 e += delta_e
56
57         if sample >= N_WAIT:
58             N_MEASUREMENTS += 1
59             U += e
60             C += e*e
61             M += abs(np.sum(spins))
62
63     U = U / N_MEASUREMENTS
64     C = beta*beta * (C / N_MEASUREMENTS - U*U)
65     M = M / N_MEASUREMENTS
66
```

```
    return U, C, M

if __name__ == '__main__':
    jsonData = []

    Ts = np.arange(0.2, 4.1, 0.2)
    #for i_N, N in enumerate([10, 50, 100]):
    for i_N, N in enumerate([10, 50, 100]):
        print("N =", N)
        _mmc = partial(mmc, N=N)

        U, C, M = [], [], []
        with Pool(processes=8) as pool:
            for vU, vC, vM in pool.map(_mmc, Ts):
                U.append(vU)
                C.append(vC)
                M.append(vM)

        jsonData.append({'N': N, 'T': list(Ts), 'U': U, 'C': C, 'M': M})

    with open("ising_2d_data.json", "w") as outFile:
        json.dump(jsonData, outFile)
```

Program for 2D Ising model (plotting)

```
import json
import math

import numpy as np
import matplotlib.pyplot as plt

def M_theory(beta):
    beta_C = 1 / (2 / math.log(1 + math.sqrt(2)))
    if beta < beta_C:
        return 0
    else:
        x = math.sinh(2*beta)
        return math.pow(1 - 1/(x*x*x*x), 1/8)

plt.rc('text', usetex=True)
f, axarr = plt.subplots(3, 1, figsize=(15,10))

for i in range(3):
    axarr[i].set_xlabel(r"$T$")
axarr[0].set_ylabel(r"$U/N^2$")
axarr[1].set_ylabel(r"$C/N^2$")
axarr[2].set_ylabel(r"$M/N^2$")

with open("ising_2d_data.json") as inFile:
    jsonData = json.load(inFile)

for data_set in jsonData:
    N = data_set["N"]
    Ts = np.array(data_set["T"])
    U = np.array(data_set["U"])
    C = np.array(data_set["C"])
    M = np.array(data_set["M"])
```

```
33
34      label = "N = {}".format(N)
35      axarr[0].plot(Ts, U/N/N, label=label)
36      axarr[1].plot(Ts, C/N/N, label=label)
37      axarr[2].plot(Ts, M/N/N, label=label)
38
39  axarr[2].plot(Ts, np.vectorize(M_theory)(1/Ts), label="Theory")
40
41  for i in range(3):
42      axarr[i].legend()
43
44  plt.savefig("ising_2d.pdf")
45  plt.show()
```