Übung 8
Computational Physics III

Matthias Plock (552335) Paul Ledwon (561764)

5. Juli 2018

Inhaltsverzeichnis

1 Monte-Carlo-Simulation: Magnetisierung

Neben der Messung der Magnetisierung wurde auch die Wirkung S gemessen. Es sollten die Autokorrelationszeiten fuer die Wirkung und die Magnetisierung verglichen werden, diese lagen bei ungefaehr 2 und 5 Monte-Carlo-Zeitschritten.

Die Definition $\sigma^2 = \frac{2\tau_{int}}{N}\Gamma$ wurde ueberprueft, jedoch unterliegt diese Abweichungen von bis zu zwei Groessenordnungen.

Die benoetigte Monte-Carlo-Zeit bis zur Thermalisierung war nie geringer als ungefaehr 100, nach der Thermalisierung wurden nocheinmal 1000 sweeps durchgefuehrt. Damit ist die Bedingung, dass die Auto-Korrelationszeit viel kleiner sein muss als diese Groessen in guter Naeherung erfuellt.

Fuer die verschiedenen L wurde jeweils fuer die Punkte mit nichtverschwindender Magnetisierung ein Fit mit der Funktion $M(\kappa) = A\sqrt{\kappa - \kappa_c}$ durchgefuehrt, da aus der Vorlesung dieser Zusammenhang in der Naehe von κ_c bekannt war. Dies ist in Abb. 1 dargestellt. Mit steigendem L scheinen die Messpunkte besser dem theoretischen Zusammenhang zu folgen, da ein hoeheres L einer hoeheren Aufloesung des Systems entspricht, ist dies auch zu erwarten. Fuer $L = 16$, also die hoechste hier getestete Aufloesung erhaelt man aus dem Fit

$$\kappa_c = 0.2553 \pm 0.0005$$

2 Planung auf der GPU

Bei der Implementierung der MC-Simulation auf der GPU sollen die sweeps auf der GPU ausgefuehrt werden und das Ergebnis der Messung gleich auf den Host kopiert werden. Hierzu werden folgende Kernel benoetigt

random_cnfg Erstellt zufaellige Spinstartkonfiguration

set_groups Teilt alle Punkte des Gitters auf zwei Gruppen auf. Bei der Einteilung auf die Gruppen kommt es darauf an, dass ein Punkt nicht in einer Gruppe mit seinen naechsten Nachbarn liegt. Dies erfolgt in 2 Dimensionen nach einem Schachbrettmuster bzw. in d Dimensionen nach dem Kriterium ob die Summe aller Koordinaten x_i eines Punktes gerade oder ungerade ist.

randgpu Erzeugt die benoetigten Zufallszahlen

alocal_G1 Berechnet die lokale Wirkung der Punkte in einer Gruppe

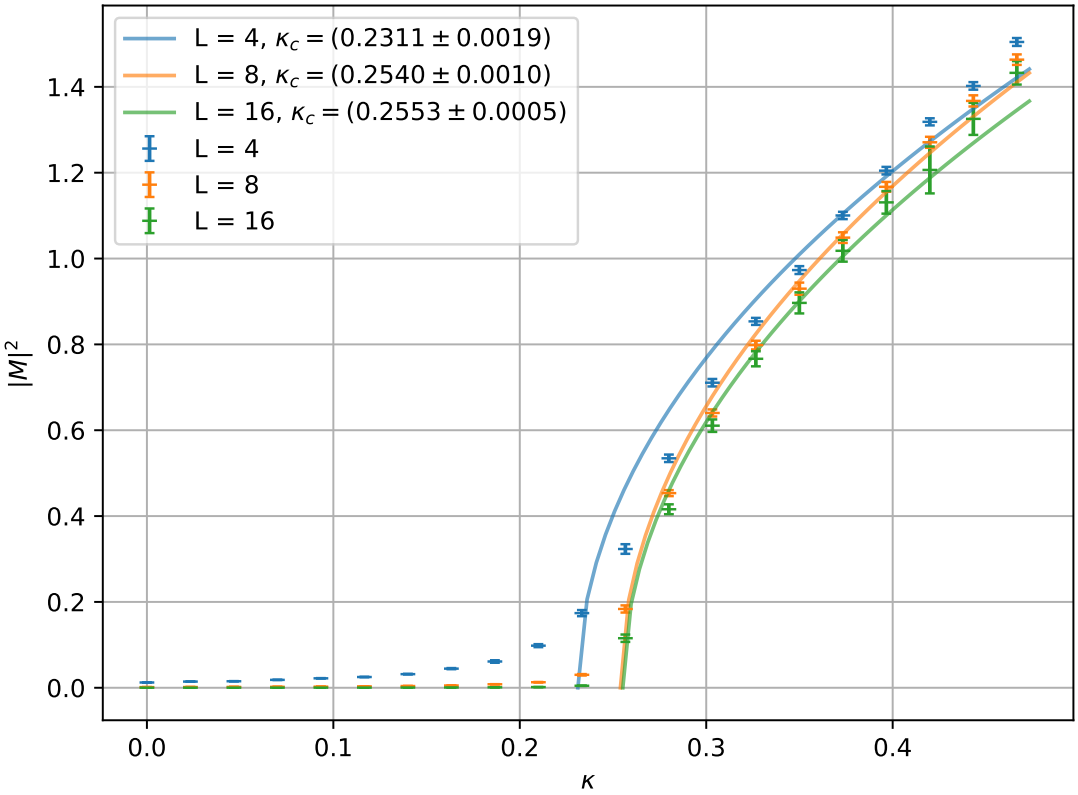

Abbildung 1: Magnetisierung fuer verschiedene L und Bestimmung von κ_c

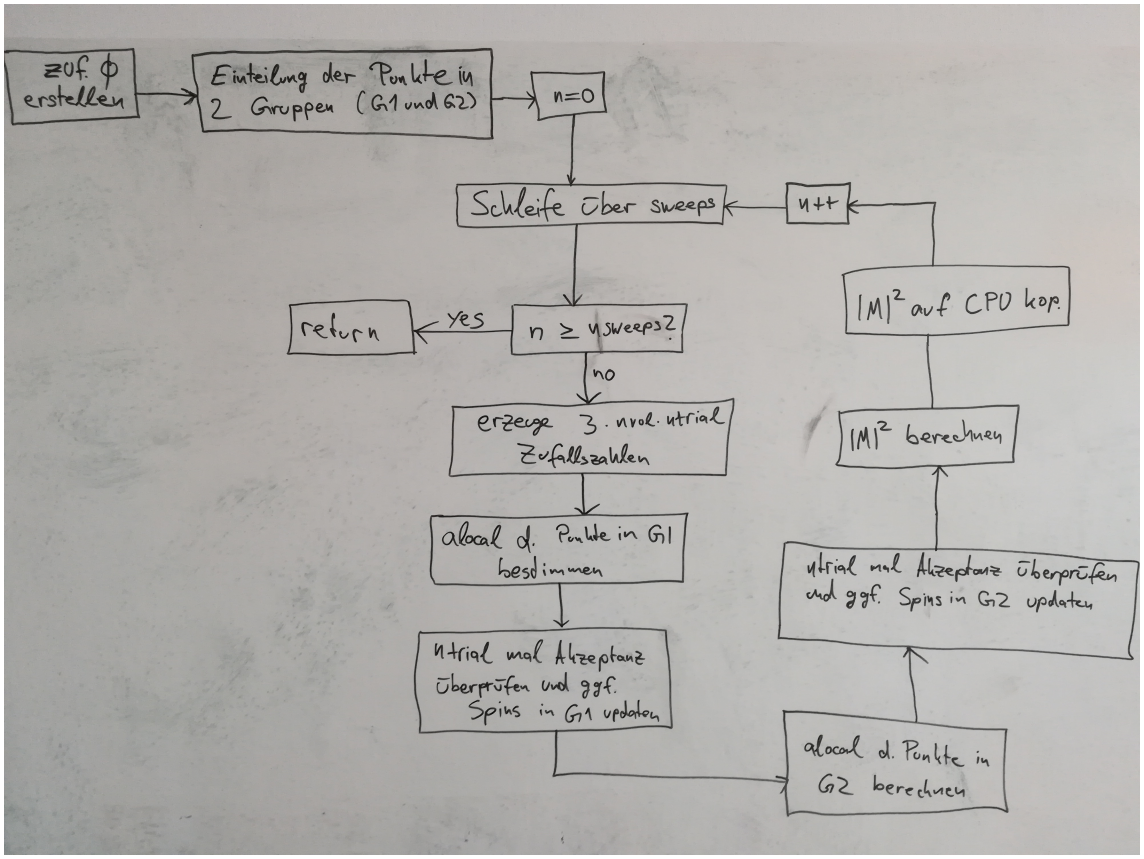

Abbildung 2: Flowchart zur Implementierung der MC-Simulation auf der GPU

spin_update Berechnet Akzeptanzwahrscheinlichkeit fuer jeden Punkt einer Gruppe und updated den Spin entsprechend dieser Wahrscheinlichkeit. Dies wird fuer jeden Punkt $ntrial$ mal durchgefuehrt

magnetisierung Aus dem Spinvektor wird die Magnetisierung mit Hilfe eines Reduce-Unrolling-Algorithmus berechnet

Ein Flowchart fuer die Implementierung auf der GPU ist in Abb. 2 dargestellt. Fuer die Ausführung werden gleichzeitig immer nur $nvol/2$ Threads benötigt, da nur die Hälfte der Daten gleichzeitig bearbeitet wird.