Übung 9
Computational Physics III

Matthias Plock (552335) Paul Ledwon (561764)

13. Juli 2018

Inhaltsverzeichnis

1 Monte-Carlo-Simulation auf der GPU

Der Algorithmus der mittels einer Monte-Carlo-Simulation die Magnetisierung eines d-dimensionalen Spinsystems berechnet, wurde auf der GPU implementiert. Hierbei mussten alle Spins in zwei Gruppen derart aufgeteilt werden, dass sie nicht mit ihren nächsten Nachbarn in einer Gruppe sind. Für diese beiden Gruppen wurde nun nacheinander ein sweep durchgeführt. Mit den gleichen Zufallszahlen (und der gleichen Spinstartkonfiguration) die für den Algorithmus auf der GPU genutzt wurden, wurde auch ein Sweep auf der CPU durchgeführt. Hierbei musste auch auf der CPU eine Einteilung der Spins in die selben Gruppen wie auf der GPU durchgeführt werden, da die Reihenfolge in der die Spins aktualisiert werden einen Einfluss auf die Messung hat.

Trotz dieser Maßnahmen ist das Ergebnis von beiden Messungen nicht identisch. Nach einem Sweep erhält man folgende Werte

```
./run Starting...
Gittergroesse: 32 x 32 x 32

nsweep = 1
lambda = 1.000000
kappa  = 1.000000
delta  = 0.200000
h      = 0.000000 + I 0.000000
phi    = random


Initalize spins... 0.026365 sec.

Using Device 0: Quadro K4000
ITER    ACC          ACC_C     DELTA         DELTA_C      GPU_MM      CPU_MM
1    0.917346      0.900763    0.210000      0.210000     0.002316    0.000005
cpu: 0.013127s, gpu: 0.002363s
```

Führt man mehrere Sweeps durch, so werden zwar für den Algorithmus auf der GPU und CPU die Alzeptanz um den gewünschten Wert von 0.4 fluktuieren und δ wird konstant, jedoch erhält man unterschiedliche Magnetisierungswerte, was eigentlich nicht passieren sollte. Ursachen hierfür sind möglicherweise, dass durch einen Implementierungsfehler doch nicht die gleichen Zufallszahlen benutzt wurden, oder dass eventuell die Konstanten $\lambda, h, \kappa, ..$ nicht korrekt auf die GPU kopiert wurden, da Probleme mit der Funktion `cudaMemcpyToSymbol` auftraten.

Den Speedup S der durch die Parallelisierung erreicht wird, kann man aus dem Output eines runs berechnen.

$$S = \frac{t_{\text{CPU}}}{t_{\text{GPU}}} \approx 5.55$$