Übungsaufgaben zur Vorlesung

Computerphysik

Priv.-Doz. Dr. R. Bulla

SS 2012

Blatt 12: Abgabetermin: Montag, der 09.07.2012, in der Vorlesung;

Aufgabe 1: Metropolis-Algorithmus für das Ising-Modell

Das Ising-Modell für eine eindimensionale Kette der Länge N (offene Randbedingungen) ist gegeben durch

$$H = -J \sum_{<ij>} s_i s_j - h \sum_i s_i = H(\{s_i\}) \ ,$$

mit $\sum_{<ij>}$ der Summe über nächste Nachbarn der Kette und mit den klassischen Spin-Variablen $s_i = \pm 1$. Mit folgendem Programm soll eine Markov-Kette aus Spin-Konfigurationen $\{s_i\}^l$, $l = 0, 1, \ldots, M-1$, erzeugt werden:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define M 10
#define N 4

main()
{
  int i, k, s[N], snew[N];
  double Eold, Enew, DeltaE, J, h;
  double alpha, beta, gamma;

  J = 1.0;
  h = 0.0;
  beta = 1.0;

  srand(time(0));

  /* Teilaufgabe a: die Start-Konfiguration */
  ...
  /* Ende Teilaufgabe a */

  for (i = 0; i < M; ++i) {

    /* Teilaufgabe b: ein zufaellig ausgewaehlter Spin
```

```
        wird umgedreht */
    ...
    /* Ende Teilaufgabe b */

    /* Teilaufgabe c: Berechnung des Energieunterschieds DeltaE
       zwischen neuer (-> Enew) und alter Spin-Konfiguration
       (-> Eold) */
    ...
    /* Ende Teilaufgabe c */

    /* Teilaufgabe d: der Metropolis-Algorithmus*/
    ...
    /* Ende Teilaufgabe d */

    /* Ausgabe der aktuellen Spin-Konfiguration */
    for (j = 0; j < N; ++j)
      printf("%d ",s[j]);
    printf("\n");

 }

  return 0;
}
```

Dazu sind noch die folgenden Programmteile zu ergänzen:

a) Die Start-Konfiguration (gespeichert im Feld `s`) soll zufällig gewählt werden. Verwenden Sie dazu die `rand()`-Funktion.

b) Für die neue Spin-Konfiguration (gespeichert im Feld `snew`) wird ein zufällig ausgewählter Spin umgedreht.

c) Berechnen Sie den Energieunterschied `DeltaE` zwischen neuer ($\rightarrow$ `Enew`) und alter Spin-Konfiguration ($\rightarrow$ `Eold`).

d) Dieser Programmteil enthält den eigentlichen Metropolis-Algorithmus: die Änderung wird mit Wahrscheinlichkeit α akzeptiert, mit

$$\alpha = \frac{w(\{\bar{s}_i\})}{w(\{s_i\}^l)} = \exp\left(-\beta(H(\{\bar{s}_i\}) - H(\{s_i\}^l))\right)$$

e) Wie berechnet man die Erwartungswerte für die Magnetisierung $m = \sum_i s_i$ und der Energie mit Hilfe des Metropolis-Algorithmus?

Hinweis: Abrunden auf ganze Zahlen geschieht in C mit `floor`, also `floor(3.1415)` ergibt 3.

Aufgabe 2: Zustandssumme des Ising-Modells

Die Zustandssumme des Ising-Modells ist gegeben durch

$$Z = \sum_{\{s_i\}} e^{-\beta H(\{s_i\})} \ ,$$

mit H definiert in Aufgabe 1. Die Summe geht dabei über alle 2^N möglichen Spin-Konfigurationen. Jede Spin-Konfiguration $\{s_i\} = (s_1, s_2, \ldots, s_N)$ lässt sich durch eine Integer-Variable j mit $0 \leq j \leq (2^N - 1)$ kodieren, und zwar durch

$$j = \sum_{i=1}^{N} n_i 2^{i-1} \quad \text{mit} \quad n_i = \frac{1}{2}(s_i + 1) \in \{0, 1\} \ .$$

Mit dieser Kodierung genügt eine einzige `for`-Schleife für die Summe $\sum_{\{s_i\}}$, in dieser Schleife muss jedoch das j noch in die Sequenz $(n_1, n_2, \ldots, n_N)$ zerlegt werden. Das geschieht mit dem folgenden einfachen Algorithmus, der für $N = 4$ und $j = 6$ wie folgt abläuft:

* bilde $j - 2^{N-1} = 6 - 2^3 = -2 < 0$;
  $\Rightarrow n_4 = 0$;
* bilde $j - 2^{N-2} = 6 - 2^2 = 2 \geq 0$;
  $\Rightarrow n_3 = 1$ und j wird auf $j = 6 - 2^2 = 2$ gesetzt;
* bilde $j - 2^{N-3} = 2 - 2^1 = 0 \geq 0$;
  $\Rightarrow n_2 = 1$ und j wird auf $j = 2 - 2^1 = 0$ gesetzt;
* bilde $j - 2^{N-4} = 0 - 2^0 = -1 < 0$;
  $\Rightarrow n_1 = 0$;

Das folgende Programm soll die Zustandssumme durch exakte Aufsummation aller Beiträge berechnen.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define N 5

main()
{
  int i, l, k, lprime, M;
  int ni[N], s[N];
  double sum, E, J, h, beta;

  J = 1.0;
  h = 0.0;
  beta = 1.0;

  sum = 0.0;
  M = (int) pow(2,N);
```

```
  for (k = 0 ; k < M; k++) {
    l = k;

  /* Teilaufgabe a: Zerlegung von l in die Sequenz
     ni[0], ni[1], ...
  ...
  /* Ende Teilaufgabe a */


  /* Ausgabe dieser Sequenz */
  for (i = 0; i < N; ++i)
    printf("%d",ni[i]);

  printf("\n");

  /* Umrechnung ni -> s */
  for (i = 0; i < N; ++i)
    s[i] = 2*ni[i] - 1;

  /* Teilaufgabe b: Berechnung des Beitrags zur Zustandssumme

  /* Ende Teilaufgabe a */


  }

  printf("Zustandssumme = %f\n",sum);

  return 0;
}
```

Dazu sind noch die folgenden Programmteile zu ergänzen:

a) Zerlegung von `l` in die Sequenz `ni[0], ni[1], ... ni[N-1]`.

b) Berechnung des Beitrags zur Zustandssumme.