Computergestütztes wissenschaftliches Rechnen
SoSe 2004

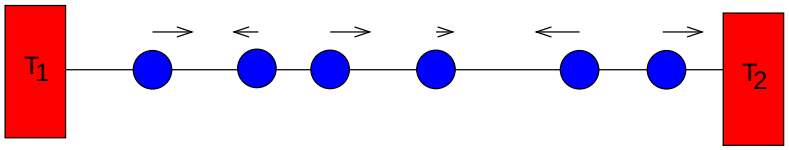
Alexander K. Hartmann, Universität Göttingen

10. Juni 2004

7 Ereignisgesteuerte Simulationen

7.1 Eindimensionale Kette harter Teilchen

Modell: Kette von harten Teilchen i mit Masse m_i, Ort x_i, Geschwindigkeit v_i



Wände bei $x = 0/x = L$ mit Wärembädern (Temperatur T_1/T_2).
Wechselwirkung der Teilchen $i.i+1$: idealer Stoß (vorher v_i, nachher v_i')

$$\begin{aligned} v_i' &= \frac{m_i - m_{i+1}}{m_i + m_{i+1}} v_i + \frac{2m_{i+1}}{m_i + m_{i+1}} v_{i+1} \\ v_{i+1}' &= \frac{2m_i}{m_i + m_{i+1}} v_i - \frac{m_i - m_{i+1}}{m_i + m_{i+1}} v_{i+1} \end{aligned} \tag{1}$$

Wechselwirkung mit Wänden:
Temperatur gemäß "Maxwell-Verteilung" verteilt [1]

$$P_{1/2}(v) = \theta(\pm v) \frac{mv}{T} \exp(-mv^2/2T_{1/2}) \tag{2}$$

Zufallszahlen gemäß $P(v)$ leicht mit Inversionsmethode auszulosen, da Verteilungsfunktion leicht integrierbar.

Ziel: Untersuchung des Wärmetransports zwischen den Bädern.

7.2 Ereignisse

Algorithmus:
NICHT: schrittweise Integration (MD-Simulation) der Bewegungsgleichung (zu langsam).
SONDERN: Teilchen bewegen sich mit v =const, bis zum nächsten Stoß: → Änderungen nur bei Stößen (=Ereignis).

Für jedes Teilchen: speichere Ort $x_i(t_i)$ bei letzter Kollision t_i

$$\begin{aligned} x_i(t) &= x_i(t_i) + (t - t_i)v_i \\ x_{i+1}(t) &= x_{i+1}(t_{i+1}) + (t - t_{i+1})v_{i+1} \end{aligned} \tag{3}$$

Beim Stoß: $x_i(t^*) = x_{i+1}(t^*) \Rightarrow$
Stoßzeit:

$$t^* = \frac{(x_i(t_i) - t_i v_i) - (x_{i+1}(t_{i+1}) - t_{i+1}v_{i+1})}{v_2 - v_1}$$

7.3 Implementierung

Vorüberlegung:
Datenstrukturen: *System, Teilchen, Ereignisse*
Funktionen: *Initialisierung, Berechung der Stoßzeit, Ausführung eines Stoßes, Auslosung einer Geschwindigkeit, Auswertung, Hauptschleife*

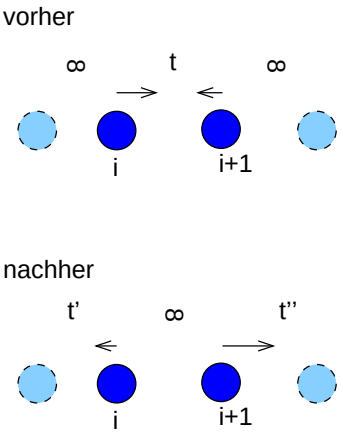
System:
Größe, Teilchenzahl, Temperaturen, Zeit, maximale Zeit.
siehe Typ `global_t` in `chain.c`

Teilchen:

```
/** data structure for one particle **/
typedef struct
{
  double  m;                              /* mass */
  double  x;       /* position at last collision*/
  double  t;          /* time of last collision */
  double  v;   /* velocity after last collision */
} particle_t;
```

Initialisierung: Teilchen gleichmäßig zwischen $x = 0$ und $x = L$ verteilen, Geschwindigkeiten zufällig in $[-1, 1]$.

Ereignisse:
Ereignis i beschreibt den Stoß zwischen Teilchen i und $i+1$. Stoßzeit ="∞", falls kein Stoß.



```
/** event= particle p1 hits on particle **/
/** p2 at time t                        **/
typedef struct
{
  double   t;                  /* time of event */
} event_t;
```

(Zunächst nur die Stoßzeit, wir später noch erweitert.)
Es wird immer das *nächste* Ereignis ausgeführt → man muß alle Ereignisse durchsuchen und das mit der kleinsten Zeit finden (SPÄTER: bessere Implementierung mit *Heap*).
Abarbeitung eines Ereignisses:
Beim Ereignis i werden Ereignisse $i-1$ und $i+1$ (Sonderfall Wände) neu ausgerechnet, neue Stoßzeit für Ereignis $i =$ "∞.
Routine `treat_event()`

```
/******************** treat_event() *****************/
/** Treat event 'ev' from 'event' array:            **/
/** calculate new velocities of particles ev,ev+1   **/
/** recalculate events ev-1, ev, ev+1               **/
/** PARAMETERS: (*)= return-paramter                **/
/**       glob: global data                         **/
/**       part: data of particles                   **/
/**      event: array of events                     **/
/*          ev: id of event                         **/
/** RETURNS:                                        **/
/**    nothing                                      **/
/*****************************************************/
void treat_event(global_t *glob, particle_t *part, event_t *event, int ev)
{
  int pl, pr;               /* particles of collision */
  double vl, vr;           /* velocities of particles */

  pl = ev;
  pr = ev+1;

  part[pl].x += (event[ev].t-  part[pl].t)*part[pl].v;
  part[pr].x += (event[ev].t - part[pr].t)*part[pr].v;
  part[pl].t = event[ev].t;
  part[pr].t = event[ev].t;

  if(pl==0)                 /* collision w. left wall */
  {
    part[pr].v = generate_maxwell(part[pr].m, glob->T1);
    event[pl].t = glob->t_end+1;
    event[pr].t = event_time(pr, pr+1, glob, part);
  }
  else if(pr==(glob->n+1)) /* collision w. right wall */
  {
    part[pl].v = -generate_maxwell(part[pl].m, glob->T2);
    event[pl].t = glob->t_end+1;
```

```
    event[pl-1].t = event_time(pl-1, pl, glob, part);
  }
  else
  {
    vl = part[pl].v; vr = part[pr].v;
    part[pl].v = ( (part[pl].m-part[pr].m)*vl + 2*part[pr].m*vr )/
      (part[pl].m + part[pr].m);
    part[pr].v = ( 2*part[pl].m*vl - (part[pl].m-part[pr].m)*vr )/
      (part[pl].m + part[pr].m);
    event[pl-1].t = event_time(pl-1, pl, glob, part);
    event[pl].t = glob->t_end+1;
    event[pr].t = event_time(pr, pr+1, glob, part);
  }
}
```

Achtung: möglicherweise zeitweise KEIN Ereignis für ein Teilchen (weder Stoß rechts noch links), ist aber kein Problem.

7.4 Dichte

Meßgröße: Dichte als Funktion des Ortes. (auch möglich: Wärmeleitung etc) Realisierung:(`glob.L`= Größe des Systems)

```
  double *density;             /* for measuring rho(x) */
  int bin, num_bins;
  double delta_x;

  ...

  num_bins = 50;
  delta_x = glob.L/num_bins;
  density = (double *) malloc(num_bins*sizeof(double));
  for(bin=0; bin<num_bins; bin++)
    density[bin] = 0;
```

Messung (`part[p]`= Daten für Teilchen `p`, `glob.n`= Anzahl der Teilchen):

```
      for(p=1; p<=glob.n; p++)
      {
        bin = (int) floor(
           (part[p].x+(t_measure-part[p].t)*part[p].v)/
           delta_x);
        density[bin]+= 1/delta_x;
      }
```

Aufbau der Hauproutine. Grobplanung durch *Pseudocode*

algorithm main()
begin
    Initialisierung
    t =erstes Ereignis
    while $t < t_{\text{end}}$
    begin
        Messungen;
        bearbeite Ereignis;
        t =nächstes Ereignis
    end
end

(siehe `main()` in `chain.c`)

$m_i = m \, \forall i$ ist trivial:
Teilchen tauschen Impulse aus
$\Leftrightarrow$ Teilchen fliegen durcheinander durch
$\Leftrightarrow$ keine Wechselwirkung

Hier: alternierende Massen ($m_1 = 1/m_2 = 2.6$)
$n = 100$ Teilchen, Laufzeit $t_{\text{end}} = 100$. Messung der Dichte nach der Hälft der Laufzeit alle 10 Zeiteinheiten. System noch nicht equilibriert:

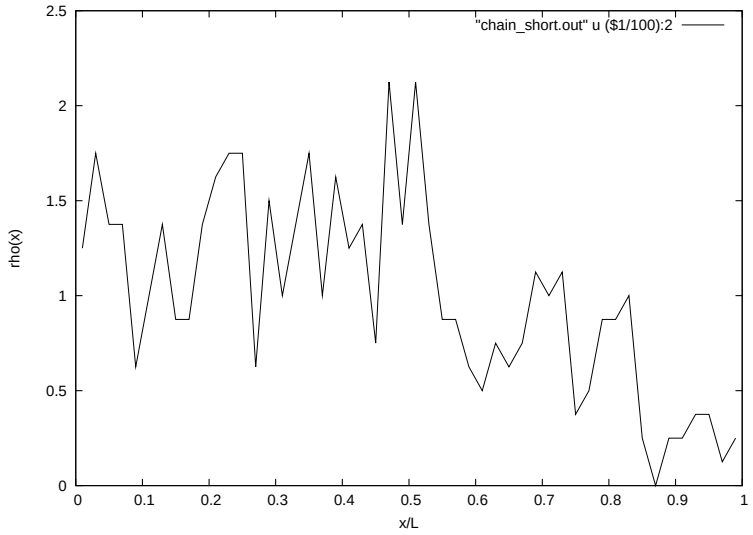


Abbildung 1: mittlere Dichte als Funktion des Ortes im Zeitintervall $[50, 100]$.

$t_{\text{end}} = 10000$.

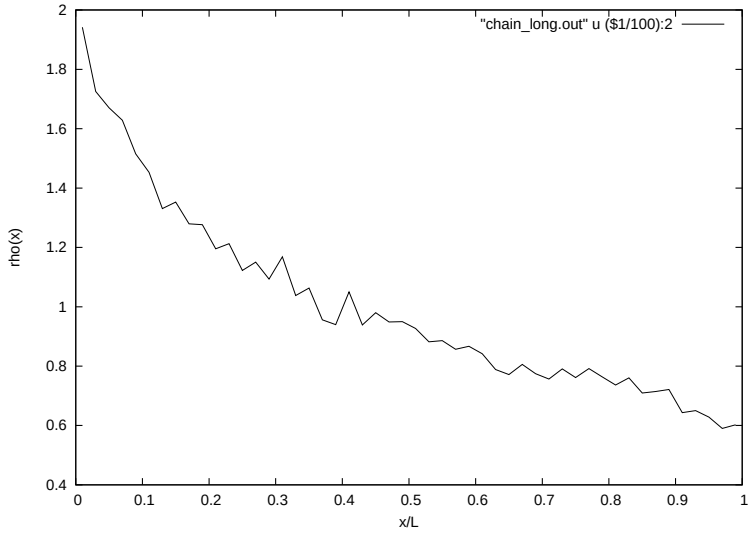


Abbildung 2: Mittlere Dichte als Funktion des Ortes im Zeitintervall $[5000, 10000]$.

Dicht geringer, dort wo die Temperatur höher ist. Weitere Ergebnisse siehe [1].

Literatur

[1] A. Dhar, Phys. Rev. Lett. **86**, 3554 (2001)