Vorlesung über Simulationsmethoden in der Physik (Computerphysik II) WS2012/13
Dr. Gerd Schröder-Turk mit Dipl.-Phys. Matthias Hoffmann

Blatt 2 – Event-Driven MD & anomale Diffusion in porösen Medien

Zur Bearbeitung in den Übungsstunden bis zum 11. Januar 2013
Letzter Abgabetermin für Feedback: 18. Januar 2013

Aufgabe 1: Billard-Tisch per event-driven MD ohne Event-Kalendar

Programmieren Sie eine Simulation eines idealisierten Billard-Spiels, d.h. 15 harte Scheiben in einer rechteckigen Domäne der Größe $L_x \times L_y$ ($L_x > L_y$ mit elastischen Stößen zwischen den Teilchen und zwischen der Wand ("Bande") und den Teilchen. Wenn die Teilchen beim Stoß mit der Wand näher als ein Schwellwert dr an eine der sechs "Taschen" kommen (also die Ecken des Tisches bzw. die Mittelpunkte der längeren Banden, so sollen die Teilchen dort verschwinden. Zur Lösung dieser Aufgabe soll eine event-driven Molekulardynamik-Simulation ohne Event-Kalendar verwendet werden.

(a) Simulieren Sie zunächst die Bewegung eines einzelnen Teilchens, indem Sie den nächsten Stosszeitpunkt mit einer der Wände berechnen und das Teilchen dann bis zu diesem Zeitpunkt gradlinig fortbewegen. An den Wänden benutzen Sie dann die bekannten Gesetze für den elastischen Stoß.

(b) Führen Sie beim Stoß eine Abfrage ein, die das Hereinfallen in die Taschen bearbeitet.

(c) Erweitern Sie nun Ihre Simulation auf mehrere Teilchen.

(d) (Zusatzaufgabe, zum Profilieren) Machen Sie Ihre Simulation noch etwas realistischer, indem Sie nicht mehr elastische Stöße sondern Stöße mit Energiedissipation annehmen. Hinweise zu diesem Verfahren finden Sie im Buch von Rapaport oder im Buch von T. Pöschel, "Computational Granular Dynamics: Models and Algorithms", Springer, unter dem Stichwort "restitution coefficients". Erkären Sie bei der Präsentation die Idee hierbei.

(e) (Zusatzaufgabe, zum Profilieren) Welche Ansätze gibt es, in event-driven MD-Algorithmen die Reibung zwischen Billardtischfilzboden und Kugeln zu simulieren?

Aufgabe 2: Billard-Tisch per event-driven MD mit Event-Kalendar

Bearbeiten Sie diegleiche Aufgabe wie in Aufgabe 1, allerdings diesmal einem Even-Kalendar. Sie können sich dabei an der Anleitung im Buch von Rapaport halten, oder Sie betrachten die Hinweise, die auf der Rückseite dieses Blattes gegeben werden und einer Aufgabe aus dem letzten Jahr entsprechen. Vergleichen Sie quantitative die unterschiedlichen Laufzeiten.

Aufgabe 3: Paar-Korrelationsfunktion von harten Kugel-Fluiden

(a) Verändern Sie das Programm aus Aufgabe 2 so, daß es ein Ensemble aus $N \gg 1$ harten Kugeln in einer Simulationsbox mit periodischen Rändern simuliert.

(b) Benutzen Sie Ihre Simulation um nach Anzeichen für einen Übergang zwischen einer diffusiven ungeordneten flüssigen Phase zu einer geordneten Phase ohne (oder mit viel kleinerer Diffusion) zu suchen, als Funktion der Dichte. Geben Sie eine Schätzwert für den Volumenanteil, bei dem dieser Übergang stattfindet an.

(c) Suchen Sie in der Literatur nach Informationen über Existenz und Art dieses Phasenübergangs.

(d) Berechnen Sie die Paarkorrelationsfunktion dieses Systems und diskutieren Sie deren Konsistenz mit der Erwartung.

Aufgabe 4: Perkolation im Booleschen Modell für poröse Medien in 2D

(a) Das poröse Medium soll durch ein "Boolesches Modell" von überlappenden Kugeln mit Radius $R = 1$ dargestellt werden. Dazu legen Sie eine Zahl M_K an Scheiben fest, und plazieren diese an zufällig ausgewählten Koordinaten $(x, y) \in [0, L]^2$, wobei L die Größe der Simulationsbox L ist. Die Zahlen x, y sind zufällig in $[0, L]$ gleichverteilte Zahlen. Beachte, dass beliebiger Überlapp zwischen den Teilchen möglich ist. Erzeugen Sie einige Konfiguration und stellen diese grafisch dar. (Bemerkung: In der mathematisch strikten Version des Booleschen Modells ist der relevante Parameter die Intensität ρ. Dann variiert die Gesamtteilchenzahl M_K zwischen unterschiedlichen Realisierungen, genaugenommen folgt Sie einer Poisson-Verteilung $\rho/M_K! \exp(-\rho)$).

(b) Bestimmen Sie die Perkolationseigenschaften: Ein *Cluster* ist eine zusammenhängende Gruppe von überlappenden Scheiben. Ein *perkolierendes* Cluster ist ein Cluster, bei dem mindestens eine Kugel mit dem oberen Rand der Simulationsbox $y = 0$ und mindestens eine Kugel mit dem unteren Rand $y = L$ Überlapp hat. Bestimmen Sie die Wahrscheinlichkeit $P(M_K, L)$, dass es ein perkolierendes Cluster gibt als Funktion von M_K für einige ausgewählte $L = L_0, 2L_0, 4L_0, ...$ Recherchieren Sie dieses Phänomen mittels des Schlagwortes "Kontinuumsperkolation" und erkären Sie das Verhalten.

(c) Diskutieren Sie, wie die Perkolationseigenschaften der Struktur aus überlappenden Scheiben (in 2D) mit den Perkolationseigenschaften des Komplements, also des Porenraumes, zusammenhängt. Wie könnte man das Voronoi-Diagramm benutzen, um die Perkolationseigenschaften des Porenraumes numerisch auch direkt zu bestimmen?

Aufgabe 5: Anormale Diffusion – Lorentz-Modell

In dieser Aufgabe ist ein einzelnes balistisches Teilchen mit Radius σ auf den Porenraum des Booleschen Modells (d.h. das Komplement der M_K Kugeln) beschränkt. Dieses Teilchen erfährt an den M_K Kugeln elastischen Stoss.

(a) Berechnen Sie die Trajektorien dieses Teilchens für eine festgewählte Realisierung des porösen Mediums. Visualisieren Sie diese.

(b) Diskutieren Sie die Trajektorien eines Systems mit Radien (R, σ) und eines mit Radien $(R+\sigma, 0)$ für die Scheiben des porösen Mediums und des ballistischen Teilchens. Unterscheiden Sie sich?

(c) Diskutieren Sie die Abhängigkeit der Trajektorie von dem Betrag $|v_0|$ der Anfangsgeschwindigkeit des Teilchens.

(d) Untersuchen Sie das Diffusionsverhalten des Teilchens, d.h. $\delta^2(t) = \langle [r - r(t=0)]^2 \rangle$. Hierbei soll $\langle \cdot \rangle$ die Mittelung über verschiedene Realisierungen der Anfangsposition des balistischen Teilchens für eine (oder wenige Realisierungen des porösen Mediums). Wählen Sie fix $|v| = 1$. Was passiert mit dieser Kurve, wenn Sie σ verändern?

(e) Diskutieren Sie anhand von $\delta^2(t)$ mit verschiedenen σ die Regime, in denen gewöhnliche Diffusion oder lokalisierte Bewegung beobachtet wird. Untersuchen Sie, ob Sie in diesem Modell sogenannte anormale Diffusion, d.h. ein Potenzgesetz $\delta^2(t) \propto t^{2/z}$ mit $z \neq 2$ finden können und ob dieses mit der Perkolation des porösen Mediums verknüpft ist?

(f) Recherchieren Sie in der Literatur, was das Lorentzmodell für anormale Diffusion ist.

Anleitung für die Programmierung des Event-Kalendars (I)

Betrachten Sie ein einzelnes Teilchen der Masse $m = 1$ und Radius $R = 1$ in einem rechteckigen Container $G = [0, L] \times [0, H]$ mit periodischen Randbedingungen in der horizontalen Richtung bei $x = L$ und harten Wänden bei $y = 0$ und $y = H$. Mit Boden und Decke soll das Teilchen elastisch stossen.

In dieser Aufgabe soll der Event-Kalendar programmiert werden für den einfachen Fall, daß es nur ein Teilchen gibt. Damit das Programm später verallgemeinert werden kann, sind einige Teile dennoch so ausgelegt, als ob insgesamt N Teilchen existieren würden.

(a) Programmieren Sie einen Event-Kalendar für dieses System. Dazu benutzen Sie als Darstellung für ein Ereignis eine

```
struct eventtype{
    int ida, idb;
    float t;
}
```

wobei `ida` der Teilchenindex ist (falls es sich um ein Teilchenereignis handelt), also hier immer 0, und durch `idb` der Eventtype codiert wird. Dabei soll

idb = -2	Durchschreiten der periodischen Ränder
idb = -3	Wandkollision
ida = -4 und idb = -4	Ende der Simulation
ida = -5 und idb = -5	Bildschirmausgabe

sein. Die Ereignisse speichern Sie dabei in einem Vektor

```
std::vector<eventtype> eventpool( NTeilchen+NExtraEvents );
```

wobei `NParticles` die Teilchenzahl und `NExtraEvents` die Zahl der außergewöhnlichen Ereignisse (Ende der Simulation, Bildschirmausgabe) ist. Sie unterhalten eine Liste

```
std::vector<int> unusedevents();
```

aller Einträge in diesem Vektor, die gegenwärtig nicht benutzt sind.
Den Event-Kalendar selber speichern Sie als

```
std::multimap<double,unsigned int> evcal;
```

wobei der *index* die Ereigniszeit darstellt und der *Wert* den Index des Ereignisses in der Liste `eventpool` ist. Die Speicherung als `multimap` nimmt Ihnen das Design des binären Baumes ab. Insbesondere liefert `unsigned int first = evcal.begin()->second;` den Index des nächsten anstehenden Ereignisses.

Weiterhin benötigt der Eventkalendar die folgende Funktionalität:

```
insertInEvCal( eventype event ){
    Speicher event in pool;
    Trage index in evCal ein;
}

removeFromEvCal( int i ){ // i = index in eventpool
    Lösche aus evCal (mittels find nach t)
    Aktualisiere unusedevents
}

deleteFirstEvent(){
    int i = evCal.begin() -> second;
    removeFromEvCal(i);
}
```

(b) Schreiben Sie nun eine Molekulardynamik-Simulation des Teilchens unter Benutzung des Event-Kalendars. Implementieren Sie dazu eine Funktion

```
PredictEvent( int pid ){ // pid = Teilchenindex
    eventype evColWall = NextCollisionEvent( pid );
    eventype evPBCCros = NextPerBCrossEvent( pid );
    if( evColCont.t < evCellCro.t )
     cal.insertEvent( evColWall );
    else
     cal.insertEvent( evPBCCros );
}
```

wobei die Funktionen `NextCollisionEvent` und `NextPerBCrossEvent` das nächste Kollisions- bzw. Randdurchschreiten-Ereignis für Teilchen `pid` bestimmen. Schreiben Sie ebenfalls Funktionen `ScheduleEndOfSimulation` und `ScheduleScreenPrintEvents`. Damit können Sie insgesamt die MD-Simulation durch ein Programm

```
Predictevent( 0 )
while( ){
    earliestev = eventpool.at( evCal.first->second );
    ida = 0
    if earliestev ist Simulations-Ende
       Aktualisiere Koordinaten
       Abbruch
    if earliestev ist Durchschreiten der Ränder
       Aktualisiere Koordinaten
       PredictEvent(ida)
    if earliestev ist Bildschirmausgabe
       Aktualisiere Koordinaten
       Gib Konfiguration auf Bildschirm aus
    if earliestev ist Wandkollision
       Aktualisiere Geschwindigkeiten
       Lösche Ereignis
       PredictEvent(ida)
}
```

(c) Testen Sie Ihr Programm ausführlich mittels der graphischen Ausgabe auf dem Bildschirm für verschiedene Anfangskonfigurationen.

(d) Ergänzen Sie Ihr Programm nun zusätzlich um eine Zellunterteilung (auch wenn diese für ein Teilchen eigentlich keinen Sinn macht): Programmieren Sie eine Zellunterteilung `Cell`, die G in $N_c \times M_c$ Zellen der Größe $\Delta_x \times \Delta_y = L/N_c \times H/M_c$ unterteilt. Die Liste `Cell[i][j]` ist eine Liste aller Teilchen, die sich gegenwärtig in Zelle `(i,j)` befinden, d.h. deren Koordinaten $(x, y) \in [i\Delta_x, (i+1)\Delta_x) \times [j\Delta_y, (j+1)\Delta_y)$ sind. Schreiben Sie eine Funktion `updateCell[i, x', y']`, die die Zuordnung von Teilchen i aktualisiert, wenn sich dessen Koordinaten zu `x', y'` verändert haben. Ergänzen Sie den Eventkalendar um das Ereignis `Zellübertritt`.

Anleitung für die Programmierung des Event-Kalendars (II)

Betrachten Sie das gleiche System wie zuvor, allerdings sollen nun N Teilchen vorhanden sein, die elastisch miteinander stossen und sich ansonsten nicht beeinflussen.

Erweitern Sie Ihr Programm für diesen Fall. Beachten Sie dazu die folgenden Hinweise:

(a) Ergänzen Sie die Funktion `PredictEvent`, sodaß sie zusätzlich das nächste Teilchen-Teilchen-Kollisions-Ereignis berücksichtigt. Dazu benötigen Sie eine Funktion `NextParticleCollisionEvent( pid )`, die insbesondere die Information in `Cell` benützt.

(b) Für die `struct eventtype` sollen nun Werte `ida` $\geq$ `0` und `idb` $\geq$ `0` einen Teilchen-Teilchen-Stoss bedeuten. Für ein Ereignis soll das Teilchen `ida` das `primäre` Teilchen und `idb` das sekundäre Teilchen heissen. Für jedes Teilchen `id` darf es zu jedem Zeitpunkt im Event-Kalendar nur ein einziges Ereignis geben, bei dem `id` das primäre Teilchen ist, und dieses Ereignis ist das nächste Ereignis, an dem Teilchen `id` beteiligt ist.

(c) Wenn das Ereignis ein Teilchenstoss zwischen `ida` und `idb` oder ein Wandstoß von Teilchen `ida` ist, müssen nun alle Ereignisse im Kalendar gelöscht werden, die Stöße mit diesen Teilchen sind. Ausserdem müssen für alle Teilchen `id`, die bei einem dieser gelöschten Ereignisse das primäre Teilchen waren, die Funktion `PredictEvents(id)` aufgerufen werden. Warum müssen keine neuen Ereignisse berechnet werden für Teilchen, die sekundär an einem Ereignis beteiligt waren? Konstruieren Sie ein Beispiel!

(d) Um den Zugriff auf alle Ereignisse, an denen ein Teilchen beteiligt war, zu bewerkstelligen, müssen Sie Ihren Kalendar erweitern um eine Liste

```
std::vector<std::list<unsigned int> > involving;
```

die so implementiert ist, daß `involving[i]` eine Liste aller Ereignisse in `eventpool` ist, an denen Teilchen `ida` oder `idb` beteiligt sind. Sie müssen nun insbesondere die Funktionen `insertInEvCal( eventype event )` und `removeFromEvCal` erweitern. Die Funktion `removeFromEvCal( int i)` soll nun *alle* Ereignisse löschen, an denen die beiden Partner von Ereignis `i` beteiligt waren. Sie sollte ausserdem eine Liste `touchedParticles` aller derer Teilchen, die als primäres Teilchen an einem Stoss mit einem der Partner von Ereignis i beteiligt waren, zurückgeben. Bei der Bearbeitung aller Stoßereignisse sollte dann für alle Teilchen in dieser Liste und für die Teilchen `ida` und `idb` die Funktion `PredictEvent` aufgerufen werden.

(e) Implementieren Sie für jedes Teilchen eine persönliche (oder lokale) Zeit.

(f) Überprüfen Sie insbesondere, daß es nach jedem Schritt für jedes Teilchen *genau* ein Ereignis geben muss, bei dem das Teilchen Primärpartner ist.