Simulazione Montecarlo di un ensamble NPT di Hard Boxes
14 Gennaio 2021

Francesco D'Amico

1 Teoria

1.1 Metropolis Montecarlo

Nell'ambito dello studio di sistemi di particelle interagenti, per i quali è possibile scrivere una Hamiltoniana di interazione, gli algoritmi di Metropolis Montecarlo sono un ottimo strumento per lo studio delle proprietà di equilibrio.
Se il sistema è all'equilibrio, prese due configurazioni j e k, tra queste vale la condizione di bilancio dettagliato:

$$\rho_j \pi_{jk} = \rho_k \pi_{kj} \tag{1}$$

dove le ρ sono le probabilità che all'equilibrio il sistema si trovi nelle due configurazioni, e le π le probabilità di transizione tra le due configurazioni. Gli algoritmi di questo tipo funzionano attuando un "walking" nello spazio delle configurazioni ρ_i, che avviene in accordo con l'Eq. 1. Pertanto, le mosse che avvengono durante la simulazione non sono fisiche: è un movimento astratto, compiuto lungo una traiettoria nello spazio delle configurazioni. La scelta di Metropolis consiste nello scegliere, per fare una mossa dalla configurazione "o" (old) ad una "n" (new):

$$\pi_{on} = \alpha(o \to n) acc(o \to n) \tag{2}$$

dove α consiste nello scegliere una configurazione n in modo casuale, ma in modo simmetrico, cioè tale che sia $\alpha(o \to n) = \alpha(n \to o)$, mentre l'acceptance è la probabilità di accettare tale mossa proposta casualmente:

$$acc(o \to n) = min\{1, \frac{\rho_n}{\rho_o}\} \tag{3}$$

Si dimostra che la scelta di Metropolis soddisfa l'Eq. 1. Quindi, seguendo questa regola, partendo da una configurazione iniziale, si esplorano gli stati del sistema. Calcolando le osservabili nelle configurazioni esplorate, è possibile ottenere la loro distribuzione nel sistema in analisi.
Ci sono però delle complicazioni di cui tener conto: il tempo necessario alla termalizzazione del sistema, la sua ergodicità, la taglia finita e la complessità dell'algoritmo. Di queste cose si discute nella Sez. 2, nell'implementazione dell'algoritmo.

1.2 Metropolis NPT

Nell'ensamble NPT, a variare tramite mosse Montecarlo sono le posizioni e le orientazioni delle particelle, e il volume del sistema. Poiché c'è piena libertà di scegliere la grandezza $\alpha(o \to n)$, purché sia simmetrica, si sceglie di campionarla da una distribuzione uniforme. Che sia l'entità dello spostamento, della rotazione, o dell'espansione o compressione del volume, si prende un valore casuale uniforme $\in [-\eta, +\eta]$, dove η è l'entità massima che si può proporre. Per ciascun tipo di mossa si usa un valore η differente, di cui si discute più nel dettaglio nella Sez. 2.2.2 riguardante le ottimizzazioni.
Per quanto riguarda l'acceptance, dallo studio della ρ di equilibrio dell'ensamble NPT, si ottiene

$$\rho_{NPT} \propto V^N exp(-\beta U_N - \beta PV) \tag{4}$$

e quindi la scelta di Metropolis, per una mossa di volume:

$$acc(o \to n) = min\{1, \frac{\rho_{NPT}(n)}{\rho_{NPT}(o)}\} \tag{5}$$

Invece, nel caso di mosse di traslazione e rotazione, il volume rimane costante, pertanto l'eq. 5 si riduce a

$$acc(o \to n) = min\{1, e^{-\beta \Delta U}\} \tag{6}$$

Nel caso in esame, analizziamo hard boxes, ovvero particelle "dure" a forma di parallelepipedo, che possono essere immaginate come punti circondati da un potenziale infinitamente repulsivo. I semiassi di tali parallelepipedi vengono scelti come a=2, b=1, c=1. Poiché per particelle dure non ci sono potenziali di interazione a distanza, il calcolo di ΔU si riduce al semplice controllo di eventuali overlaps risultanti dalla mossa proposta, poiché sono possibili solamente i due valori $e^{-\beta \Delta U}$=$\{0,1\}$.

2 L'algoritmo

L'algoritmo si compone delle seguenti fasi:

1. Si fissano i parametri fisici N, P, T, che saranno costanti nella simulazione, il parametro fisico iniziale V, che cambierà a seguito delle mosse di volume, e tutti quei parametri utili all'ottimizzazione dell'algoritmo.

2. Si esegue il monte carlo step: per N-1 volte si propone una mossa di traslazione o rotazione di una singola particella, e poi una volta si propone una mossa di volume.

3. Si itera il monte carlo step per un numero fissato di volte, nel caso presente 10^3. Durante questa prima fase, si aggiustano i parametri delle $\alpha(o \to n)$, affinché circa il 30-50 % delle volte la mossa sia accettata.

4. Si itera il monte carlo step per una quantità di passi fissata, senza cambiare più i parametri delle $\alpha(o \to n)$. Dopo che il sistema ha termalizzato, si comincia a campionare la grandezza di interesse, in questo caso la densità volumica delle particelle:

$$\Phi = \frac{v_{box}}{V_{simul}}$$

Grazie a questo algoritmo, otteniamo un campionamento della grandezza Φ dalla distribuzione di equilibrio del sistema. Prendendo per ogni simulazione il valore medio, e ripetendo per varie pressioni P, si ottiene l'equazione di stato $P = P(\Phi)$.

2.1 Implementazione

La prima operazione eseguita è la preparazione della configurazione iniziale. Dati i parametri fissi N,P,T e il valore iniziale di Φ, si costruisce una configurazione iniziale ordinata, come descritto nel Listing 1.

```
void prepare_initial_conf(void){
    //contatore particelle
    int i =0;
    pvector<ntype,3> saxx;
    saxx << pars.a,pars.b,pars.c;
    //cella primitiva
    ntype num_dr;
    pvector<ntype,3> dr;
    cout << "lato = " << pars.L[0] <<"\n";
    cout << "a = " << pars.a <<"\n";
    num_dr = ceil(cbrt((float) pars.Np));
    dr[0] = 2.01*pars.a;
    dr[1] = 2.01*pars.b;
    dr[2] = 2.01*pars.c;
    //cicli su celle primitive
    for(int ix=0; ix<num_dr; ix++){
        for(int iy=0; iy<num_dr; iy++){
            for(int iz=0; iz<num_dr; iz++){
                //cout << ix << " " << iy << " " << iz<< "\n";
                if(i<pars.Np){
                    parts[i].r << ix*dr[0], dr[1]*iy, dr[2]*iz;
                    parts[i].R << 1.,0.,0.,0.,1.,0.,0.,0.,1.;
                    parts[i].set_sax(saxx);
                    //parts[i].r.show();
                    i++;
                }

            }
        }
    }
    if(i<pars.Np){
        cout << "NON SONO ENTRATE TUTTE LE PARTICELLE " << i << endl ;
        exit(1);
    }
    //creiamo da zero la linked cell list
    reset_linked_cells();

}
```

Listing 1: Si preparano le Hard-Boxes in una configurazione reticolare ordinata. I parametri sono ottimizzati per densità $\Phi < 0.2$. Infine, si crea da zero la linked cells list, ottimizzata sul volume iniziale.

Una volta pronta la configurazione iniziale, comincia la simulazione montecarlo. Come descritto in precedenza, l'ensamble NPT richiede sia mosse di spostamento (rotazione o traslazione di una particella), sia mosse di volume.

1. Le mosse di spostamento sono semplici da effettuare. Si propone con pari frequenza una traslazione o rotazione casuale, di una entità ottimizzata dalla simulazione stessa. Se, a seguito dello spostamento, la particella non ha creato overlaps, la mossa è accettata.

2. Le mosse di volume sono più difficili da implementare. Poiché una tale mossa richiede l'aggiornamento di tutte le posizioni e il conseguente controllo di tutti gli eventuali overlaps generati, la mossa impiega un tempo O(N). Pertanto si effettua tale mossa solo una volta ogni N spostamenti (che sono di O(1)), al fine di mantenere la durata totale della simulazione O(N), ed evitare che diventi O(N^2). Inoltre, anche la linked cell list deve essere rigenerata per mantenerne le caratteristiche ottimizzate al nuovo volume.

Si mostra il metodo che implementa la mossa di volume.

```
void move_box(void){
    ntype Vo, Vn, deltaV, fact;
    store_all_pars();
    bool rejected=false;
    Vo = pars.V;
    //cambiamo il volume ; pars.vmax fa in modo che il programma si aggiusti da ↩
        solo in
    //modo che l'acceptance ratio sia 30-50%
    deltaV = 2.*pars.vmax*(rng.ranf()-0.5);
    Vn = Vo + deltaV;
    //oltre a cambiare il volume dobbiamo rifare la linked cell
    //insieme a cambiare V, cambiamo anche
    fact = pars.V_change(deltaV);
    //cambiamo le posizioni
    for (int i=0; i < pars.Np; i++){
        parts[i].r = (parts[i].r)*fact;
    }
    //ricreiamo da capo la lcl
    reset_linked_cells();
    state.tot_vol +=1;
    //ora vediamo se la mossa la prendiamo con questo deltaV
    for(int i=0; i<pars.Np; i++){
        //se ci sta anche un solo overlap, è da buttare
        if(check_all_overlaps(i) == true){
            rejected = true;
            break;
        }
    }
    if(!rejected){
        ntype DG;
        DG = -(pars.betaP*(Vn-Vo)-pars.Np*log(Vn/Vo));
        //decidiamo se accettare la mossa di volume o riggetarla
        if(rng.ranf()> exp(DG)){
            rejected=true;

        }
    }
```

```
37      //se riggettiamo la mossa, bisogna:
38      if(rejected){
39          restore_all_par();
40          pars.V_change(-deltaV);
41          reset_linked_cells();
42          state.vol_rej +=1;
43      }
44  }
```

Listing 2: Funzione della mossa di volume. E' ottimizzata per avere un acceptance ratio del ≈50%. Si ricrea da capo la linked cell list per mantenerne corretto e ottimizzato l'utilizzo.

In base alla pressione fissata dall'utente, il principale dato finale che si ottiene alla fine della simulazione è il volume finale V_{simul}, da cui la densità volumica

$$\Phi = \frac{v_{box}}{V_{simul}}$$

Per le misure, sono necessarie le seguenti accortezze:

- E' necessario attendere che sia terminata l'ottimizzazione degli acceptance ratio, ovvero che questi diventino fissati. Se così non fosse, non varrebbe il bilancio dettagliato, e non è garantito che la simulazione ottenga dati fisicamente corretti.
- E' necessario attendere anche la fine della termalizzazione: il volume iniziale deve essere arrivato intorno al suo valore corretto. Per cui è importante seguire la variazione della misura in funzione del numero di steps, e fissare la soglia da cui iniziare a effettuare misure in modo opportuno.
- Il sistema, ad ogni MC step, si muove di una piccola distanza nel suo spazio delle configurazioni. Per cui, è poco utile misurare delle osservabili ad ogni step, poiché il loro valore è autocorrelato. In modo sofisticato, si potrebbe analizzare il decadimento esponenziale della funzione di autocorrelazione per valutare quanto è necessario attendere per avere misure davvero indipendenti. In modo molto più semplice, è sicuramente utile fissare un numero di MC steps fisso abbastanza grande, e prendere poche misure ma abbastanza indipendenti tra loro. Questo per due motivi: valutare meglio le fluttuazioni nelle misure e quindi gli errori, e osservare meglio il processo di termalizzazione.

2.2 Ottimizzazione

2.2.1 Le Linked Cell Lists

Senza alcuna ottimizzazione, il singolo MC step è O(N), ma è possibile utilizzare metodi di ottimizzazione che la rendono O(1). Il metodo implementato è quello della Linked Cell List. Il funzionamento è il seguente:

- Il volume è suddiviso in celle. Se l'interazione tra i centri di due particelle è al massimo a distanza Δr, allora il lato di ciascuna cella deve essere almeno Δr. Nel caso di Hardboxes, l'unica interazione possibile è quella di contatto. Per cui è certo che due particelle non interagiscano se la distanza tra i due centri è $\Delta r = d$, con d = diagonale della particella.

- Mossa una particella, per verificare se essa ora genera un overlap, è sufficiente guardare nella cella in cui si trova e nelle 26 adiacenti. E' esattamente questa caratteristica che rende il singolo MC step di O(1), ovvero non dipendente dalla taglia del sistema. Rimane però una dipendenza dalla densità, perché più questa è alta e più particelle saranno presenti in ogni cella

Anche se concettualmente semplice, l'algoritmo presenta delle difficoltà di implementazione. In particolare bisogna tenere conto di molte alternative possibili che si vengono a creare quando si fa l'update di una particella. Si procede a descrivere i punti salienti per la creazione iniziale della linked cells list:

1. Ogni particella ha due indici: il suo ("**index**"), e quello di un'altra particella (**"index next"**).

2. Si crea un array 3D, di cui ogni elemento corrisponde a una delle celle in cui è stato suddiviso il volume. Inizialmente l'array ha il valore -1 in ogni elemento.

3. Si riempie tale array 3D. Presa una particella A, si guarda la posizione dell'array che corrisponde alla cella in cui è posizionata. Se l'array in quella posizione è vuoto (cioè, ha il valore -1), si scrive sull'array il valore **index**. Se invece c'è un indice già scritto, che corrisponde ad una particella B, si guarda il suo **index next**. Se è -1, si scrive l'**index** di A come **index next** di B. Se invece l'**index next** di B è occupato da una particella C, allora si itera l'operazione: si scorre la fila di particelle fino ad arrivare ad una che ha come **index next** $= -1$. A quel punto si aggiorna questo ultimo al valore **index**.

4. Attuate queste operazioni, per ogni cella abbiamo formato una fila di particelle che si puntano ordinatamente: ora ottenere i vicini di ogni particella è diventata una operazione di complessità O(1).

Usando la linked cell list, scelta una cella, è possibile accedere a tutte le particelle contenute in questa semplicemente "scorrendo" la fila. L'altra operazione importante è l'update: ogni volta che una particella viene traslata fino ad arrivare in un'altra cella, è necessario aggiornare la linked cell list per tenere conto di questa variazione. I risultati della simulazione, con e senza utilizzo dell'ottimizzazione della linked cell list, sono riportati in Fig. 1.

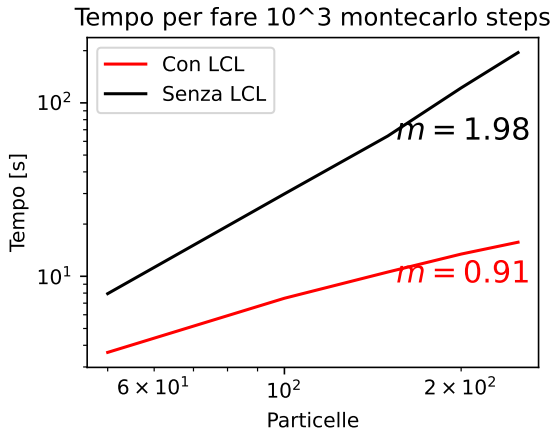


Figura 1: Comparazione dei tempi di esecuzione in scala doppio-log. Il coefficiente angolare m corrisponde all'esponente del tempo di esecuzione del programma: pertanto possiamo affermare che senza ottimizzazione, l'esecuzione è $\approx O(N^2)$, mentre con LCL è $\approx O(N)$. Per cui, quando il numero di particelle è molto grande, è indispensabile una ottimizzazione di questo tipo.

2.2.2 Il tuning dell'acceptance ratio

L'algoritmo Metropolis Montecarlo NPT è caratterizzato dalla proposta di mosse di traslazione, rotazione o di volume, che possono essere accettate o rigettate in base all'acceptance probability. Empiricamente, se la frazione di volte che un tipo di mossa viene accettata (acceptance ratio) è $\approx$ 30-50%, risulta che il sistema esplora lo spazio delle configurazioni in modo molto più rapido. Questo, qualitativamente, perchè mosse sempre accettate fanno muovere il sistema sempre, ma di una piccola entità, mentre mosse troppo grandi sono la gran parte delle volte rigettate.
Per ottenere questo risultato, si lavora sulle entità delle mosse proposte: ad esempio il valore massimo delle traslazioni proposte, o il valore massimo del fattore di compressione o espansione del volume. Questi valori cambiano molto al variare dei parametri fisici del sistema. Sono inizialmente forniti dall'utente, ma poi nei primi 10000 passi si variano gradualmente per renderli il più efficaci possibile.
E' importante ricordare che l'equazione del bilancio dettagliato non è soddisfatta variando questi parametri: pertanto i risultati ottenuti durante il tuning dell'acceptance ratio non sono corretti, ovvero non hanno un reale senso fisico.

3 Risultati

Si simulano 2000 hard boxes, per una durata totale di 10^5 monte carlo steps. Si è osservato che nell'intero intervallo delle pressioni e densità analizzate, la termalizzazione è avvenuta ad $\approx 3 \cdot 10^4$ steps. Pertanto, la soglia a partire dalla quale si comincia il campionamento dei valori di Φ è fissata a $5 \cdot 10^4$ steps. Si fissa la temperatura T=5.
L'andamento dei valori ottenuti si compara con lo sviluppo teorico del viriale al primo ordine:

$$P(\Phi) = \frac{T\Phi}{v_{hb}}(1 + \frac{v_{excl}}{2v_{hb}}\Phi) \tag{7}$$

dove $v_{hb} = 16$ è il volume della singola particella, e v_{excl} corrisponde al volume escluso di due hard boxes della forma presa in esame: per la forma del parallelepipedo è una quantità non analitica, e si ottiene numericamente che $v_{excl} \approx 192$.
In Tabella 1, e nel grafico 2 si riportano i risultati ottenuti.

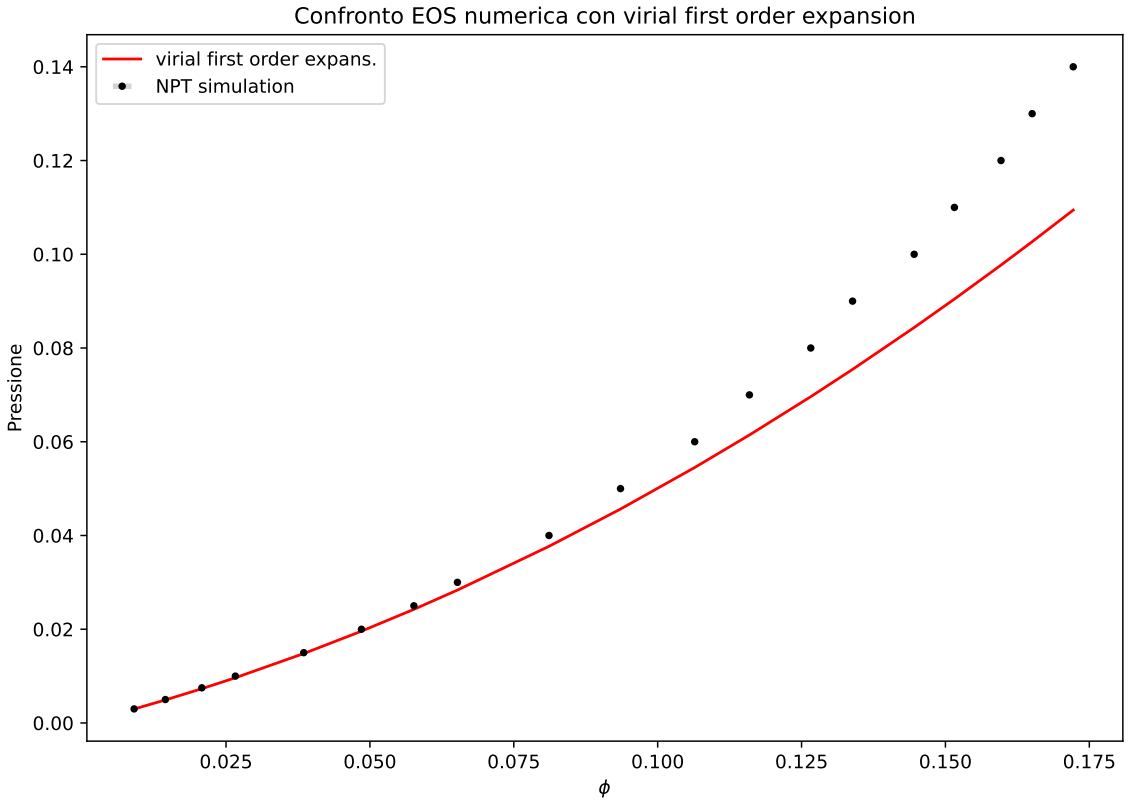
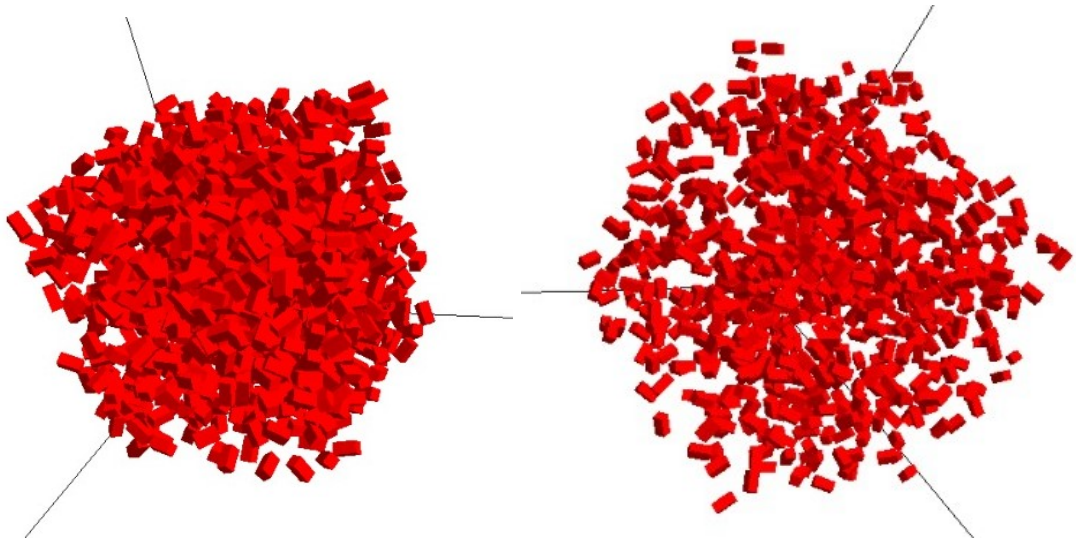


Figura 2: Comparazione tra risultati ottenuti tramite simulazione, e lo sviluppo del viriale al primo ordine. Le fluttuazioni statistiche osservate nella simulazione sono di un fattore 10^3 minori delle grandezze analizzate, pertanto non sono visibili nel grafico. Si osserva che a basse densità i punti seguono la curva teorica, mentre a densità crescenti se ne discosta. La motivazione è che a densità sufficientemente basse, qualsiasi sistema con interazioni non a lunga distanza si comporta come un gas perfetto. Aumentando la densità, in una prima fase lo sviluppo al primo ordine si accorda molto bene, ma a densità ancora maggiori l'andamento vero si discosta da quello approssimato. In particolare, si osserva che la pressione ottenuta dalla simulazione, è maggiore di quella prevista dallo sviluppo del viriale.

P	Φ	σ_Φ
0.14	0.1479	0.0004
0.13	0.1444	0.0004
0.12	0.1376	0.0003
0.11	0.1330	0.0003
0.10	0.1269	0.0003
0.09	0.1200	0.0003
0.08	0.1116	0.0002
0.07	0.1053	0.0002
0.06	0.0955	0.0002
0.05	0.0855	0.0002
0.04	0.0738	0.0001
0.03	0.0613	0.0001
0.025	0.0541	0.0001
0.02	0.0461	0.0001
0.015	0.0368	0.0001
0.01	0.0266	0.0001
0.0075	0.0207	0.0001
0.005	0.01446	0.00004
0.003	0.00902	0.00002

Tabella 1: Misure di Φ al variare della Pressione. Le incertezze σ_Φ riportate corrispondo alla deviazione standard campionaria.



(a) (b)

Figura 3: (a) Esempio di una configurazione termalizzata a P = 0.14, e densità $\Phi \approx 0.15$. (b) Esempio di una configurazone termalizzata a P = 0.03, e $\Phi \approx 0.06$.