Laboratorio di Fisica Computazionale

Zhou Zheng: matricola 873968

Indice

1 Interpolazione

Per lo studio dell'interpolazione ho preferito usare Python anziché Fortran90 per comodità personale dato che non è attualmente necessaria velocità di esecuzione del codice.

Si è costruito un programma che interpolasse prima una funzione gaussiana e successivamente una lorentziana utilizzando la costruzione di Newton in due casi: nodi equidistanziati e nodi di Chebyshev.

Sono state effettuate varie prove con diverso numero di nodi; nelle successive figure, la funzione da interpolare è la linea continua blu, mentre la funzione polinomiale interpolante è la curva tratteggiata viola; le stelle sono i nodi.

In alcuni casi è stato guardato l'errore percentuale, che è stato calcolato nel seguente modo:

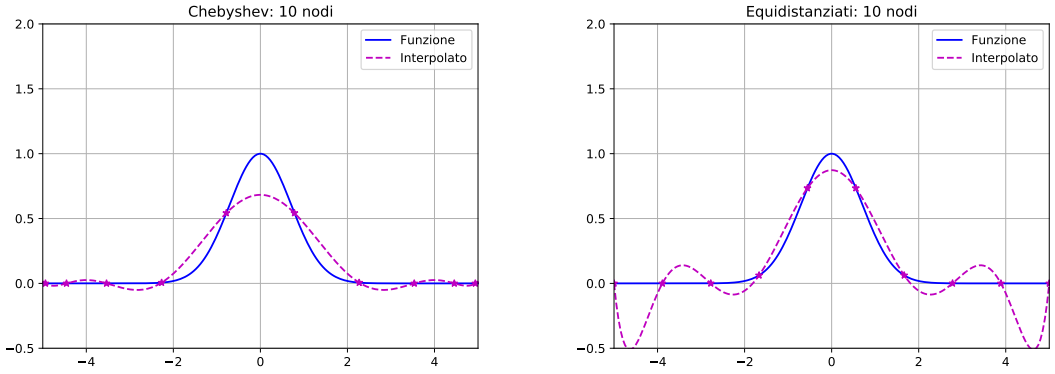
$$\text{errore percentuale } [\%] = \frac{f(x) - P_n(x)}{\max(f(x))} \cdot 100 \tag{1}$$

dove $P_n(x)$ è il polinomio interpolante di ordine n.

1.1 Funzione gaussiana

Si interpola ora la seguente funzione gaussiana:

$$f(x) = e^{-x^2} \tag{2}$$



(a) Nodi di Chebyshev

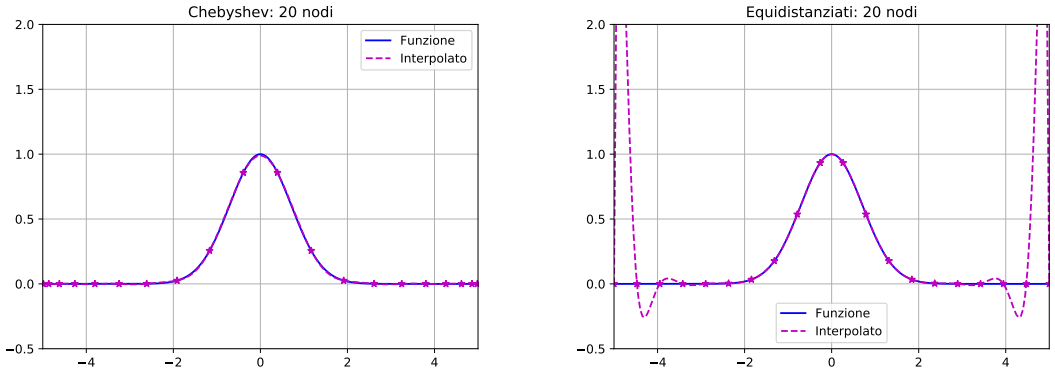
(b) Nodi equidistanziati

Figura 1: Funzione gaussiana interpolata con 10 nodi

La figura 1 mostra la curva interpolata con 10 nodi. Si nota che in nessuno dei due casi (nodi equidistanziati e nodi di Chebyshev) si riesce a generare un polinomio interpolante che descriva accuratamente l'andamento della funzione da interpolare, il numero di nodi è troppo limitato.

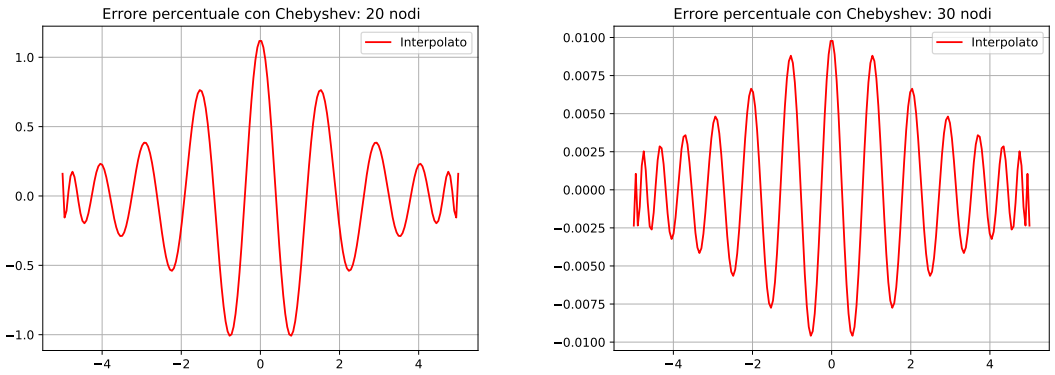
Aumentando il numero di nodi a 20 (figura 2) si ha una interpolazione migliore. Nel caso dei nodi equidistanziati (figura 2b) si ha una interpolazione ottima nel centro dell'intervallo, ma sugli estremi si notano code che divergono. Questo è causato dal fatto che gli estremi di interpolazione sono i punti più delicati e sono maggiormente soggette ad errore. Questo errore può essere compensato in due modi: aumentando il numero di nodi oppure cambiandone la distribuzione nell'intervallo,

inserendo più nodi agli estremi e meno al centro. I nodi di Chebyshev sfruttano proprio questa seconda possibilità: la densità di nodi agli estremi è maggiore rispetto alla densità di nodi al centro. La figura 2a ottenuta con nodi di Chebyshev mostra proprio un andamento senza code divergenti e con interpolazione eccellente. Nella figura 3a si legge che l'errore percentuale massimo è dell'ordine dell'1%.



(a) Nodi di Chebyshev (b) Nodi equidistanziati

Figura 2: Funzione gaussiana interpolata con 20 nodi



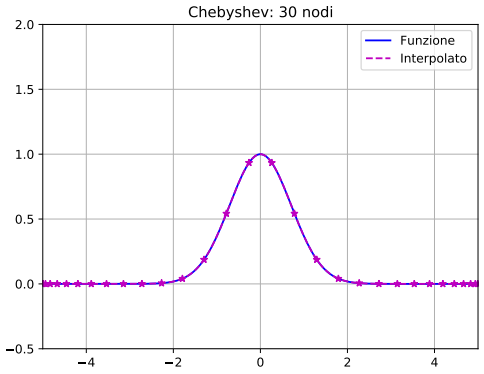
(a) Errore con 20 nodi di Chebyshev (b) Errore con 30 nodi di Chebyshev

Figura 3: Errori con interpolazione di Chebyshev

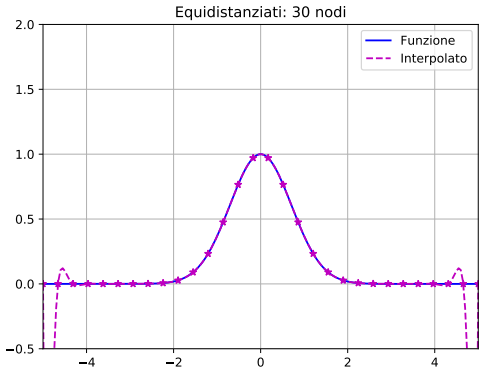
Passando a 30 nodi (figura 4) sono ancora presenti code divergenti usando nodi equidistanziati, mentre l'interpolazione con nodi di Chebyshev ha un andamento ancora migliore del caso di 20 nodi, infatti nella figura 3b si nota che l'errore massimo è dell'ordine del 0.01%.

Volendo utilizzare i nodi equidistanziati, non è nemmeno possibile aumentare troppo il numero di nodi, infatti la funzione interpolante assume un andamento non desiderato: in figura 5b si nota che le code oscillano in modo divergente (dove la funzione da interpolare è quasi nulla).

Pur non essendo necessario aggiungere più nodi nel caso di utilizzo dei nodi di Chebyshev, lo si è fatto comunque per studiare l'andamento della funzione interpolante. Si è scoperto che non bisogna

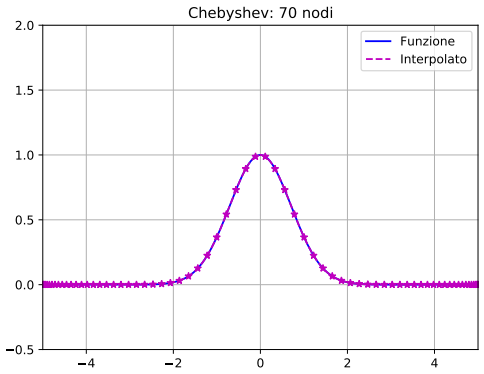


(a) Nodi di Chebyshev

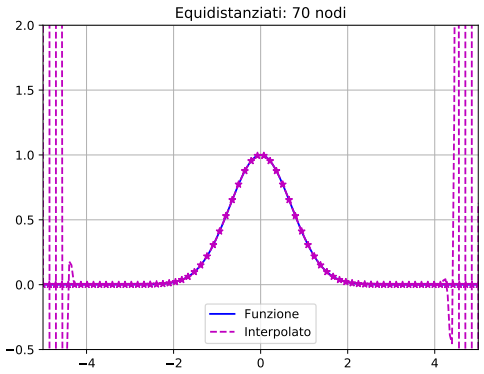


(b) Nodi equidistanziati

Figura 4: Funzione gaussiana interpolata con 30 nodi



(a) Nodi di Chebyshev



(b) Nodi equidistanziati

Figura 5: Funzione gaussiana interpolata con 70 nodi

inserire troppi nodi dato che il polinomio interpolante inizia a comportarsi in modo inaspettato (come mostrato in figura 6 con 85 nodi): nascono errori che invalidano l'interpolazione. Si suppone che questo sia dato da errori di troncamento del calcolatore dato che, matematicamente, maggiore è il numero di nodi, migliore deve essere il polinomio interpolante.

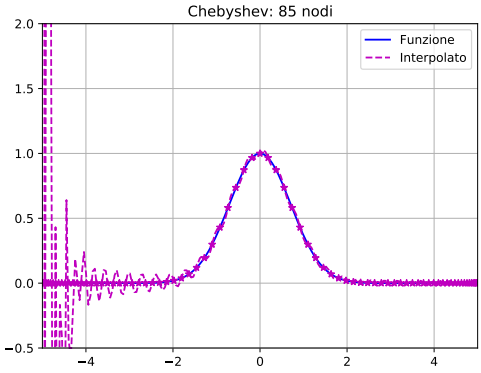


Figura 6: Funzione gaussiana interpolata con 85 nodi di Chebyshev

Una possibile soluzione alle divergenze sulle code è di interpolare su un intervallo maggiore e considerare il polinomio valido solo in un intervallo minore, ad esempio interpolare la gaussiana con 30 nodi (figura 4b) in un intervallo di $[-5; +5]$ e considerare valido solo l'intervallo $[-4; +4]$.

1.2 Funzione lorenziana

Si interpola ora la seguente funzione lorenziana:

$$f(x) = \frac{1}{1 + 25x^2} \tag{3}$$

Rispetto alla gaussiana, la lorenziana considerata ha un picco più stretto, questo si nota nella figura 7. Il fatto che abbia un picco più stretto suggerisce una più difficile interpolazione.

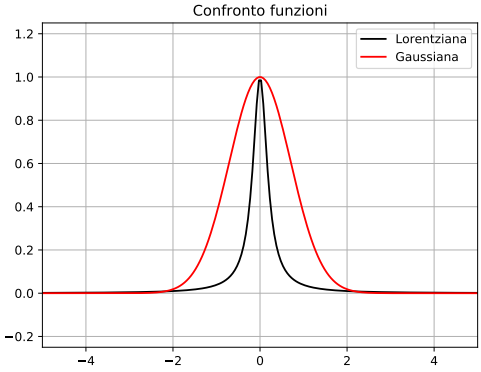
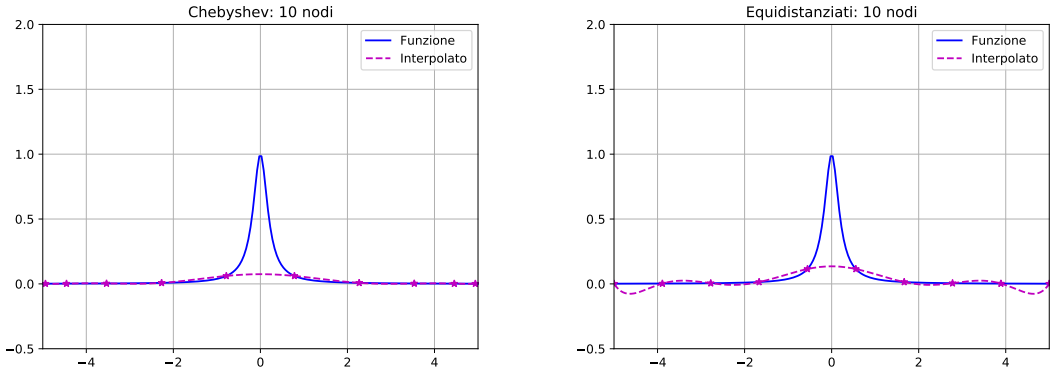


Figura 7: Confronto tra la gaussiana (2) con la lorentziana (3)

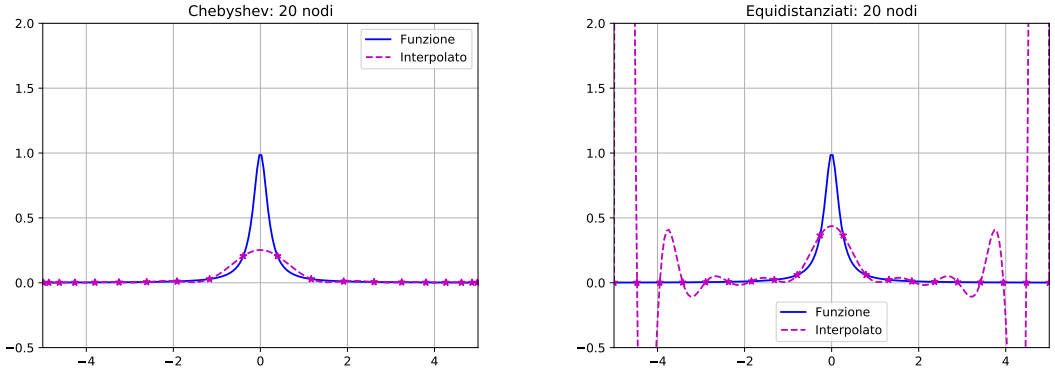
Nella figura 8 si ha un'interpolazione con 10 nodi: si nota che sono troppo pochi e non si riesce a generare il picco.



(a) Nodi di Chebyshev (b) Nodi equidistanziati

Figura 8: Funzione lorentziana interpolata con 10 nodi

Aumentando a 20 il numero di nodi, si nota dalla figura 9 che inizia a formarsi un picco leggermente più pronunciato in entrambi i casi. Tuttavia nel caso con nodi equidistanziati, si notano già delle code divergenti, questo fa supporre che le code siano legate alla presenza di picchi stretti.

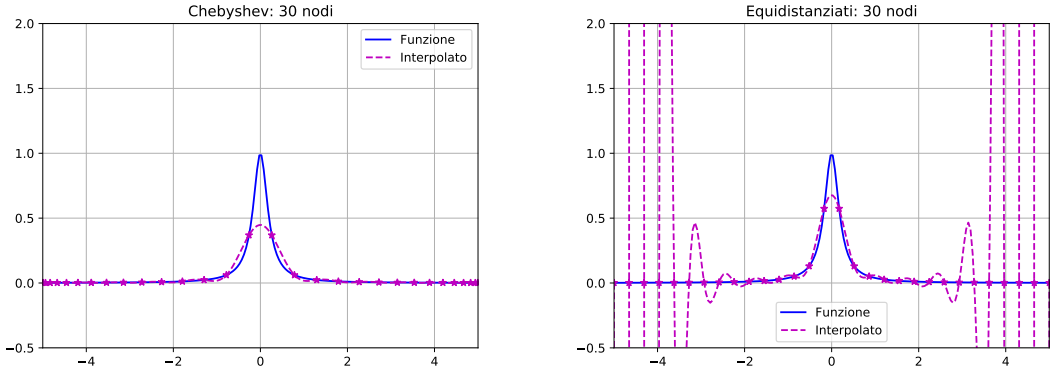


(a) Nodi di Chebyshev (b) Nodi equidistanziati

Figura 9: Funzione lorentziana interpolata con 20 nodi

Con una interpolazione a 30 nodi (figura 10) si nota un picco ancora più pronunciato in entrambi i casi. L'interpolazione con nodi equidistanziati è totalmente da scartare dato che mostra code che divergono molto. L'interpolazione con nodi di Chebyshev è leggermente migliore del caso a 20 nodi, ma ancora lontano dalla lorentziana.

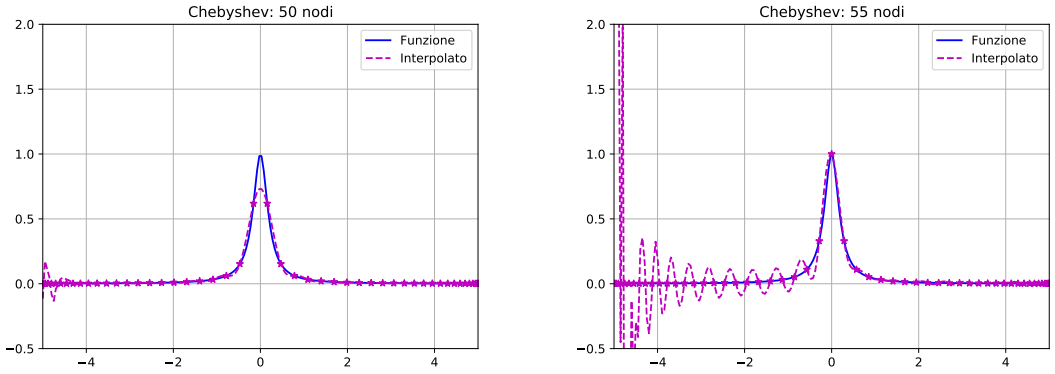
Visto l'andamento della curva interpolante con 30 nodi equidistanziati, si ritiene inutile una ulteriore analisi con più nodi. Invece il caso con nodi di Chebyshev sembra promettente, quindi si è aumentato il numero di nodi a 50 e 55, si mostrano in figura 11 i risultati. Con 50 nodi



(a) Nodi di Chebyshev (b) Nodi equidistanziati

Figura 10: Funzione lorentziana interpolata con 30 nodi

l'interpolazione del picco è migliorata, ma si iniziano a vedere errori grandi sulle code. Con 55 nodi la situazione è persino peggiore, pertanto è inutile utilizzare un numero maggiore di nodi.



(a) Funzione lorentziana interpolata con 50 nodi di Chebyshev (b) Funzione lorentziana interpolata con 55 nodi di Chebyshev

Figura 11

Una strategia per risolvere il problema nel caso di funzione lorentziana (o funzione con picco molto stretto) è di suddividere il nostro intervallo $[-5; 5]$ in più sottointervalli e trovare un polinomio interpolante per ciascun sottointervallo, in questo modo si può anche adattare il numero di nodi per ciascun sottointervallo.

2 Integrazione numerica

2.1 Regola di Cavalieri-Simpson

Si è provato ad integrare due funzioni di cui si conosce la primitiva. In particolare, si sono confrontati i valori degli integrali valutati numericamente e valutati analiticamente. Si è proceduto con numero fissato di intervalli di integrazione. Le due funzioni sono:

1. Polinomio di terzo grado:
$$f(x) = 12x^3 + 6x^2 - 8x + 1 \tag{4}$$
Ci si aspetta un errore nullo dato che la regola di Cavalieri-Simpson integra esattamente funzioni con $\frac{\mathrm{d}^3 f}{\mathrm{d}x^3} = \text{costante}$. Svolgendo l'esercizio, si è trovato proprio errore nullo.

2. Esponenziale:
$$f(x) = e^x \tag{5}$$
Non si sono trovati problemi di integrazione e si è visto che l'errore diminuisce all'aumentare del numero di intervalli di integrazione (come ci si aspettava).

Si è successivamente svolto l'esercizio proposto a lezione implementando un algoritmo adattivo e si è visto che funziona in modo accurato sulla funzione esponenziale. Si è poi applicato l'algoritmo di Simpson alla seguente funzione:
$$f(x) = \frac{1}{\sqrt{|x|}} \tag{6}$$
nell'intervallo $[-1, +1]$. L'algoritmo ha fallito nell'integrazione e il programma ha ritornato "Infinity" come risultato dell'integrale. Analiticamente si ha:
$$\int_{-1}^{+1} \frac{1}{\sqrt{|x|}}\,\mathrm{d}x = 4 \tag{7}$$
Il risultato infinito può essere dato dal fatto che la singolarità si trova in $x = 0$, quindi si prova ad eseguire il seguente integrale:
$$\int_{0}^{+2} \frac{\mathrm{d}x}{\sqrt{|x - \sqrt{2}|}} = 2\left(\sqrt[4]{2} + \sqrt{2 - \sqrt{2}}\right) \approx 4.89697448350497 \tag{8}$$
Purtroppo l'algoritmo di Simpson fallisce ancora e restituisce infinito come valore dell'integrale.

2.2 Integrazione gaussiana

Si è ripetuto l'esercizio precedente utilizzando un algoritmo che utilizzasse l'integrazione gaussiana. Si è ottenuto numericamente:
$$\int_{-1}^{+1} \frac{1}{\sqrt{|x|}}\,\mathrm{d}x \approx 3.999974, \text{ errore} = 2.4 \times 10^{-5} \tag{9}$$
$$\int_{0}^{+2} \frac{1}{\sqrt{|x - \sqrt{2}|}} \approx 4.896888, \text{ errore} = 6.1 \times 10^{-5} \tag{10}$$

Si nota che il valore dell'integrale è molto vicino al risultato analitico. Si nota inoltre che il risultato analitico si trova fuori dal raggio dell'errore, quindi il metodo utilizzato nell'esercizio per valutare l'errore non è molto preciso.

Alla luce di questi due integrali, il metodo di Simpson non è adatto a valutare integrali con singolarità integrabili, problema superato utilizzando l'integrazione gaussiana adattiva.

3 Risoluzione di equazioni differenziali

Dopo aver implementato gli algoritmi per svolgere la risoluzione numerica di equazioni differenziali con vari metodi, si è costruito un risolutore con step adattivo, ovvero un risolutore con step variabile:

1. Si è innanzitutto scelto uno step iniziale Δt_0 da cui partire e un errore massimo Δx_0, si sono impostate le condizioni iniziali t_0 e x_0.

2. Si effettua uno step di Δt_0 e si salvano $t_1 = t_0 + \Delta t_0$ e x_1.

3. Si torna ad x_0 e si effettuano due steps di $\Delta t_0/2$ ciascuno, quindi si ha $t_1' = t_0 + 2 \cdot \Delta t_0/2 = t_1$ e x_1'.

4. Si valuta l'errore come $\Delta x = |x_1 - x_1'|$ e si riscala lo step nel seguente modo:

$$\Delta t_0' = \Delta t_0 \cdot \left(\frac{\Delta x_0}{\Delta x} \right)^{1/n} \tag{11}$$

   Con $n =$ ordine dell'integratore, quindi:

   - $n = 1$ per il metodo elementare.
   - $n = 2$ per il metodo midpoint.
   - $n = 4$ per il metodo di Runge Kutta.

   In questo modo:

   - Se $\Delta < \Delta_0$ il prossimo step è maggiore.
   - Se $\Delta > \Delta_0$ il prossimo step è minore.

Si mostra in figura 12 la soluzione all'equazione seguente:

$$\dot{x}(t) = t \tag{12}$$

Si è successivamente esteso il programma in modo che comprendesse la risoluzione di un sistema di equazioni differenziali e lo si è applicato all'oscillatore armonico. Si ottiene il grafico in figura 13.

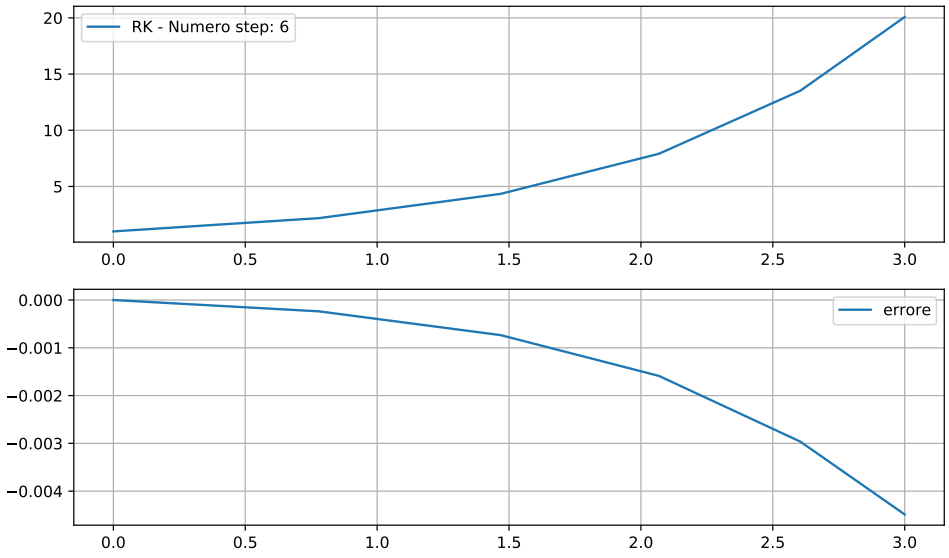


Figura 12: Soluzione all'equazione (12). L'errore è valutato sottraendo il risultato numerico alla soluzione analitica (ovvero la funzione $f(t) = e^t$). Si è chiesto $\Delta_0 = 0.01$.

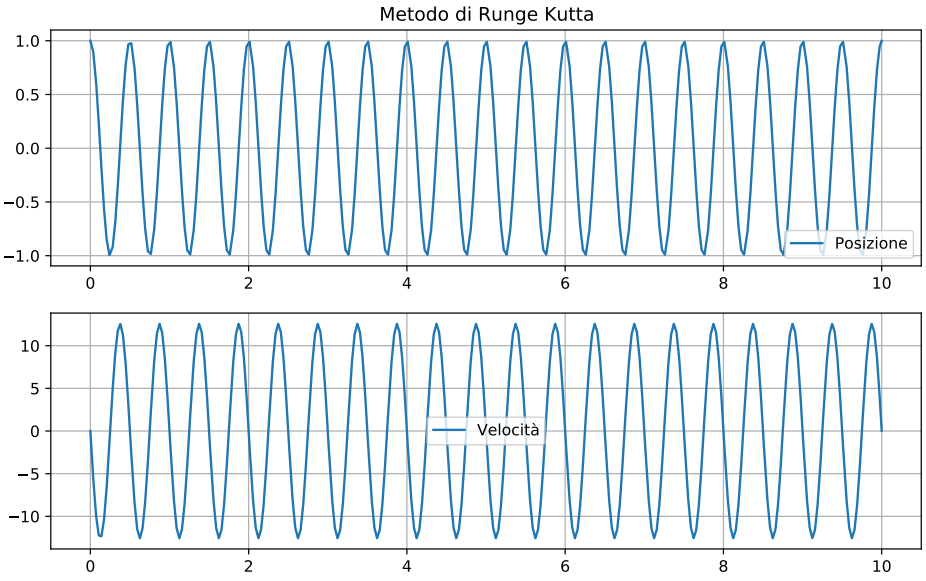


Figura 13: Oscillatore armonico con frequenza $f = 2$ ed errore massimo $\Delta_0 = 0.01$ integrato con Runge Kutta adattivo.

4 Dinamica molecolare

Ho scelto di utilizzare Fortran90 in modo da avere l'opportunità di imparare un nuovo linguaggio di programmazione e dato che è un linguaggio ottimizzato sul calcolo numerico: quando è stato inventato, il FORTRAN doveva direttamente competere con l'Assembly, quindi la velocità di esecuzione e l'efficienza era ed è fondamentale qualità del linguaggio di programmazione.

La generazione di numeri casuali si affida alla funzione di Fortran `RANDOM_NUMBER` che implementa l'algoritmo di *xoshiro256*** per la generazione di numeri pseudocasuali

L'esercizio di dinamica molecolare consiste nel simulare il comportamento di molecole monoatomiche con interazione a due corpi che segue un potenziale di Lennard-Jones (LJ) in presenza di cut-off, si ha quindi una simulazione di sfere soffici che collidono. Alla fine dell'analisi, si possono identificare queste sfere soffici con atomi di argon e confrontare la simulazione con i parametri sperimentali dell'argon stesso.

4.1 Costruzione del core del programma

4.1.1 Strutture dati

Il programma contiene una matrice `atoms` di dimensione $6 \times$ `nAtoms`: ciascuna colonna rappresenta un atomo, le prime tre righe rappresentano la posizione, mentre le ultime tre rappresentano le velocità; `nAtoms` è il numero di atomi considerati nel programma. Per comodità, ho creato un modulo Fortran da utilizzare come sezione del programma contenente variabili globali, tale modulo è il modulo *parametri*. All'aumentare della complessità del programma, ho inserito in tale modulo tutte le variabili globali richieste di volta in volta.

Le impostazioni che coinvolgono un run sono contenute nel programma principale, inoltre i vari sottoprogrammi sono stati creati in modo tale da essere più flessibili possibile, in questo modo non è stato necessario scrivere un sottoprogramma per funzionalità, ma è stato possibile adattare sottoprogrammi già presenti per soddisfare le varie esigenze. Le funzionalità sono le simulazioni per effettuare i vari studi:

1. Confronto tra integrazione con metodo di Verlet e metodo di Runge Kutta, in particolare si verifica la conservazione dell'energia.

2. Si salvano tutti gli step per ottenere un singolo punto, in questo modo si può studiare l'autocorrelazione.

3. Si trovano le curve di Van Der Waals a fissata temperatura e numero di atomi, variando il volume con la dinamica molecolare.

4. Si fonde/condensa un cristallo/gas con la dinamica molecolare, si sono poi studiati anche lo spostamento quadratico medio e la distribuzione radiale di atomi.

5. Si studia il valore di aspettazione con il metodo di Monte Carlo e l'algoritmo di Metropolis-Hastings.

6. Salva le velocità degli atomi per verificare che, raggiunto l'equilibrio, la distribuzione segua una distribuzione di Maxwell-Boltzmann.

7. Si fonde il cristallo con il metodo di Monte Carlo e l'algoritmo di Metropolis-Hastings

4.1.2 Inizializzazione

L'inizializzazione di `atoms` avviene nel file `initialization.f90`. Il volume scelto è un cubo di lato `boxLength` con condizioni al contorno periodiche.

Durante la costruzione del programma e il debug iniziale, si sono disposti gli atomi su un reticolo cubico semplice (SC), successivamente, per effettuare tutti gli studi, si sono disposti gli atomi in un reticolo cubico a facce centrate (FCC); la disposizione degli atomi avviene rispettivamente nelle funzioni `simpleCubic` e `FCC`. La velocità degli atomi è stata assegnata utilizzando coordinate sferiche e distribuzione uniforme: ovvero il modulo della velocità e le due posizioni angolari sono distribuite uniformemente; successivamente si sono riscalate queste velocità per ottenere la temperatura desiderata (variabile `requiredTemp` in *parametri*).

Il sottoprogramma `periodicConditions` implementa le condizioni al contorno periodiche: se un atomo si trova fuori dal cubo, lo si riporta all'interno.

Essendo presente questo cubo di periodicità `boxLength`, le distanze tra gli atomi andranno valutate tenendo in considerazione la periodicità, per questo si è implementato il sottoprogramma `calcDistance` (nel modulo *physics*) per valutare le distanze tra gli atomi.

4.2 Fisica

La fisica del programma si trova nel modulo *physics*. In particolare è presente il sottoprogramma `dynamics` che contiene le seguenti equazioni del moto da risolvere numericamente:

$$\begin{cases} \dot{\vec{x}}_i(t) = \vec{v}_i(t) \\ \dot{\vec{v}}_i(t) = \vec{F}_i(t) \end{cases} \quad i = 1, 2, ..., \#\text{atomi} \tag{13}$$

dove:

- $\vec{x}_i(t)$ = posizione dell'atomo i
- $\vec{v}_i(t)$ = velocità dell'atomo i
- $\vec{F}_i(t)$ = forza totale che agisce sull'atomo i

La forza è calcolata dal sottoprogramma `calcForce` nel seguente modo:

$$\vec{F}_i(t) = \sum_{j \neq i} \vec{F}_{ji}(t) \tag{14}$$

dove $\vec{F}_{ji}(t)$ = forza che il j-esimo atomo agisce sull'i-esimo atomo e deriva dal potenziale di LJ, il cut-off del potenziale è dato dal parametro `cutOff`.

Per salvare $\vec{F}_i$ si utilizza una matrice di dimensione $3 \times$ `nAtoms` chiamata `force` ed utilizzata come variabile globale all'interno del modulo *physics*.

Nel programma sono state usate unità ridotte. Dato il potenziale di LJ con cut-off:

$$u(r) = \theta(r_0 - r) 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} - \eta(r_0) \right], \; \eta(r_0) = \left(\frac{\sigma}{r_0} \right)^{12} - \left(\frac{\sigma}{r_0} \right)^{6} \tag{15}$$

si avrà:

- Unità di lunghezza: σ
- Unità di energia: ϵ
- Unità di massa: m

Inoltre si è anche posta la costante di Boltzmann $k_B = 1$.

Per verificare il corretto funzionamento del programma e per il debug, si è utilizzato il metodo del Midpoint per la risoluzione delle equazioni del moto.

Dopo aver definito la dinamica del sistema e programmato come si dovrebbe evolvere il nostro ensemble microcanonico, si definiscono le modalità per misurare le osservabili, di particolare interesse ci sono: energia, temperatura e pressione. Le costanti di conversione tra unità naturali e unità di misura del SI per l'argon sono:

- Unità di lunghezza: $\sigma = 3.4 \times 10^{-10}\,\mathrm{m}$
- Unità di temperatura: $\epsilon/k_B = 120\,\mathrm{K}$
- Unità di energia: $\epsilon = 1.656 \times 10^{-21}\,\mathrm{J}$
- Unità di pressione: $P = 4.22 \times 10^{7}\,\mathrm{Pa}$

4.2.1 Energia

L'energia è la somma dell'energia cinetica totale e dell'energia potenziale totale, quindi:

$$E_{tot} = K_{tot} + U_{tot} \tag{16}$$

dove:

- $K_{tot} = \frac{1}{2} m \sum_i v_i^2$
- $U_{tot} = \sum_{i \neq j} u_{ij}(|\vec{x}_i - \vec{x}_j|)$, $u_{ij}(|\vec{x}_i - \vec{x}_j|) =$ potenziale di LJ tra gli atomi i e j.

Dato che si ha un ensemble microcanonico, durante l'evoluzione l'energia dovrebbe conservarsi. Questo sarà una proprietà fondamentale nella scelta dell'integratore adottato.

4.2.2 Temperatura

La temperatura è una proprietà emergente, quindi non ha senso chiedersi quanto vale la temperatura di un singolo atomo. Tuttavia, dato un ensemble con abbastanza atomi, è possibile utilizzare il teorema di equipartizione dell'energia. Pertanto:

$$T = \frac{2}{3k_B} \left\langle \frac{p_k^2}{2m} \right\rangle = \frac{2K_{tot}}{3k_B N} \tag{17}$$

4.2.3 Pressione

Anche la pressione, come la temperatura, è una proprietà emergente. A differenza della temperatura però, c'è una difficoltà maggiore nella sua valutazione dato che è presente un sistema con condizioni al contorno periodiche: non si può usare la definizione usuale della pressione. Ci viene in aiuto il teorema del viriale:

$$PV = \frac{2}{3}\langle T \rangle + \frac{1}{3}\left\langle \sum_{i<j} \vec{F}^{ij} \cdot (\vec{x}_j - \vec{x}_i) \right\rangle \tag{18}$$

dove:

- $\vec{F}^{ij}$ è la forza la particella i agisce sulla particella j. Nel programma questo contributo si calcola quando si calcola la forza e viene memorizzato nella variabile globale `tempPressure`. Questo è un metodo per ottimizzare il programma dato che, quando bisogna calcolare la pressione, bisognerebbe calcolare di nuovo le forze, questo porta ad un dispendio di risorse e allungamento di tempo di calcolo. Utilizzando questo stratagemma è agevole e molto economico (in termini di tempo di calcolo) valutare la pressione a ciascuno step di integrazione. In assenza di questo metodo, bisognerebbe calcolare le forze per la risoluzione delle equazioni del moto, successivamente bisognerebbe calcolare nuovamente le stesse forze per il calcolo della pressione, aumentando di molto la durata di un run. Dopo aver finito di calcolare le forze, `tempPressure` viene valutata opportunamente per ottenere la pressione.
- $\vec{x}_j - \vec{x}_i$ è il vettore che congiunge i punti $\vec{x}_j$ e $\vec{x}_i$ tenendo conto della periodicità del cubo. Viene valutato nella funzione `calcDistance` nel calcolo della distanza.

4.3 Struttura a celle

Per un commento approfondito, si veda l'appendice A.

Per ottimizzare il programma, si è implementata una struttura a celle in modo da diminuire il numero di istruzioni eseguite: con la struttura a celle si sa a priori se una coppia di atomi potrebbe interagire, atomi presenti in celle non adiacenti non possono interagire, pertanto non si considera la coppia durante il calcolo della forza. Si è studiato il tempo di esecuzione di 10000 step di integrazione e si è trovato che il tempo di esecuzione minimo con vario numero di atomi era intorno a `m` $= 14$, dove `m` = numero di celle per lato del cubo. Si è quindi impostato `maxCells` $= 14$ come numero massimo di celle. In ogni caso, il numero massimo di celle non deve essere maggiore di `boxLength/cutOff`.

Il numero di celle e la struttura a liste è reimpostata all'inizio di ogni run se cambia `m` da un run all'altro.

4.4 Scelta dell'integratore

Per lo studio della dinamica molecolare, bisogna scegliere un integratore adatto; in particolare l'integratore deve soddisfare 2 requisiti:

1. Integratore reversibile.
2. Integratore simplettico.

I due metodi di integrazione considerati nella trattazione sono il metodo di Verlet e un integratore che implementa Runge Kutta (RK). Il metodo di Verlet è sia reversibile, sia simplettico, mentre il metodo RK non possiede nessuna delle due proprietà.

Si è preso un sistema e lo si è fatto evolvere per 1000 unità di tempo utilizzando diversi step temporali e i due metodi. Si è poi graficato la variazione di energia rispetto al sistema di partenza, ovvero si è fatto un plot con:

$$\Delta E(t) = E(t) - E(0) \tag{19}$$

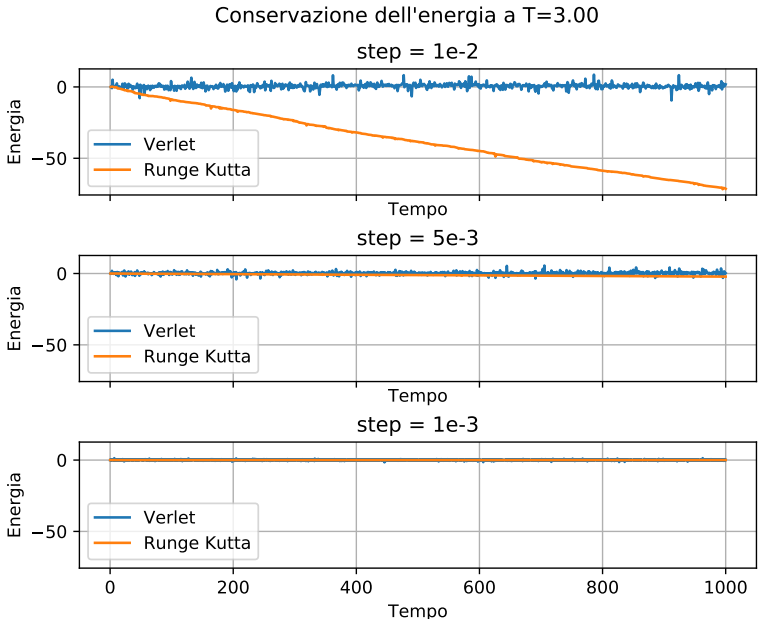


Figura 14: Valore dell'energia nel tempo con varie durate di step di integrazione, sull'asse delle ordinate è presente $\Delta E(t)$

Si mostra in figura 14 l'andamento del run. Si nota che per tutti i 3 step temporali studiati ($\Delta t = 10^{-2}, 5 \times 10^{-3}, 10^{-3}$), il metodo di Verlet conserva l'energia, mentre il metodo RK mantiene costante l'energia solo a ridotti step temporali, ma non è detto che l'energia rimanga conservata anche con run più lunghi di 1000 unità di tempo. Utilizzando il metodo di Verlet, anche con uno step di $\Delta t = 10^{-2}$ si ha errore abbastanza piccolo, quindi per basse temperature o alte temperature e grandi volumi, si usa $\Delta t = 10^{-2}$ per risparmiare tempo di calcolo (in particolare si può dimezzare il tempo di un run). A piccoli volumi con alte temperature si utilizza $\Delta t = 5 \times 10^{-3}$ per precauzione e per evitare eventuali problemi di atomi che si avvicinano troppo facendo esplodere il potenziale. Il problema appena esposto che potrebbe coinvolgere per alte temperature a volumi piccoli non si è visto durante i run per grandi temperature a grandi volumi e temperature basse in generale, mentre lo si è visto per alte temperature e piccoli volumi.

4.5 Distribuzione di velocità

Durante l'inizializzazione, le velocità sono distribuite secondo una distribuzione uniforme, come si vede in figura 15. Si è partiti da 3000 atomi dato che si costruisce in istogramma più dettagliato e da un volume molto grande: si ha un cubo di lato $L = 100$.

Già a $t = 1$ si inizia a vedere una coda che inizia a formarsi (figura 16). A $t = 3$ si hanno anche meno atomi a bassa velocità, si nota che la distribuzione inizia ad assumere la forma della distri-

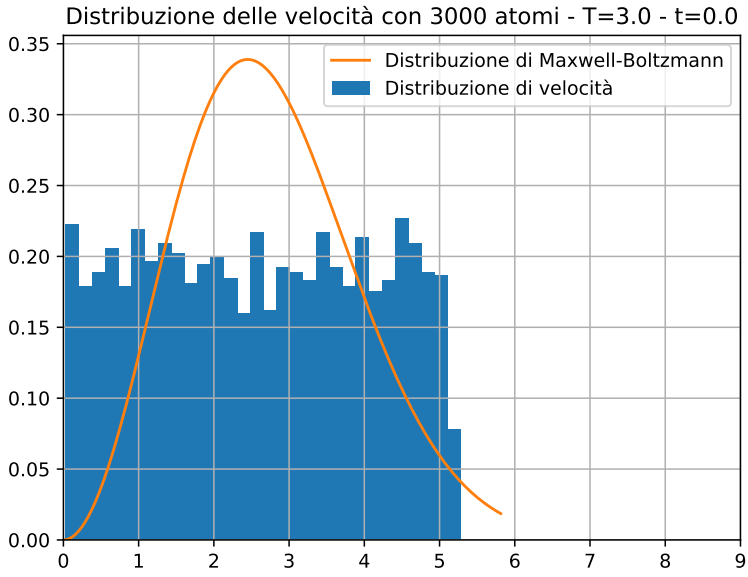


Figura 15: Distribuzione di velocità iniziale di un sistema con 3000 atomi.

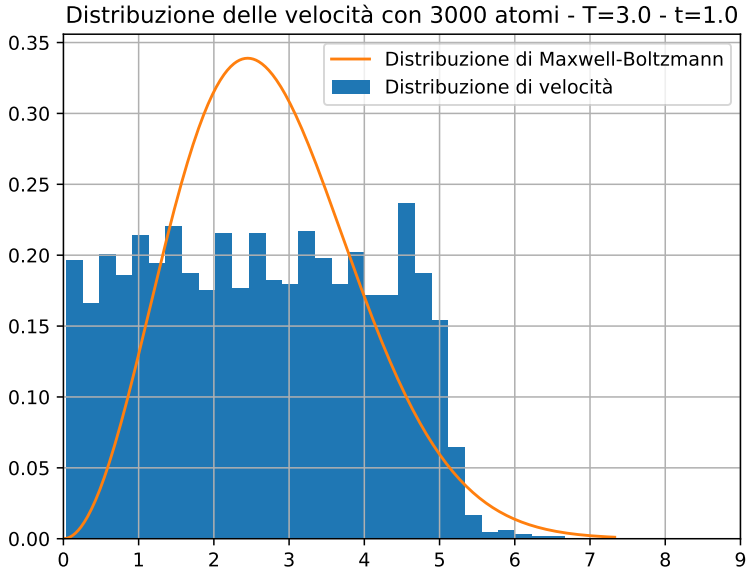


Figura 16: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=1.0.

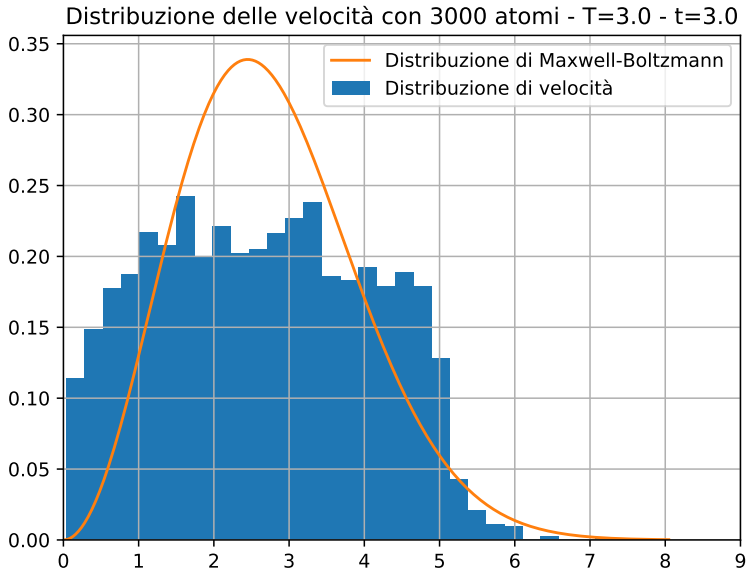


Figura 17: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=3.0.

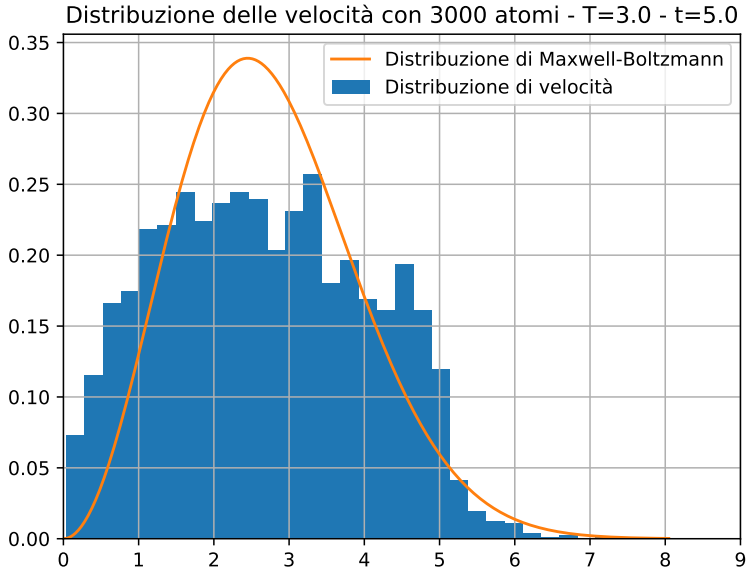


Figura 18: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=5.0.

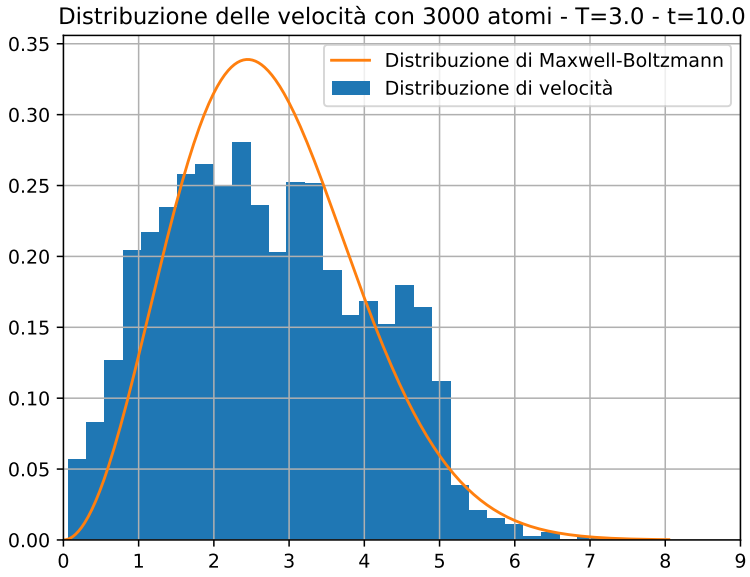


Figura 19: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=10.0.

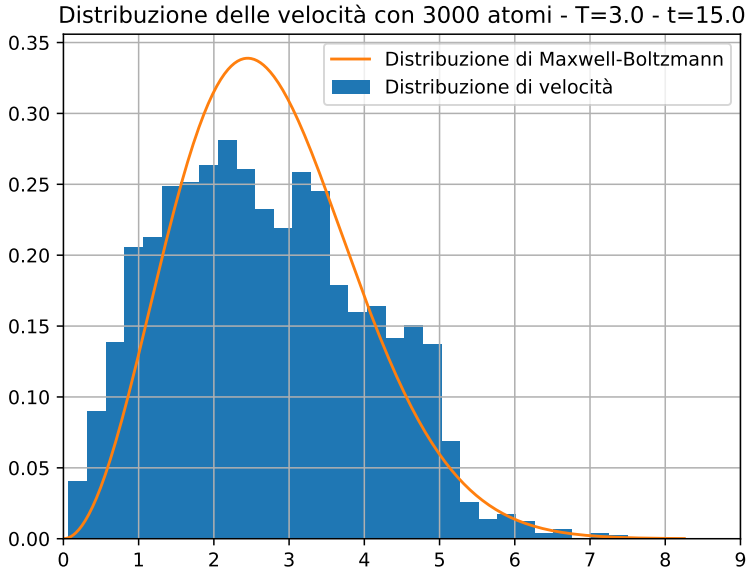


Figura 20: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=15.0.

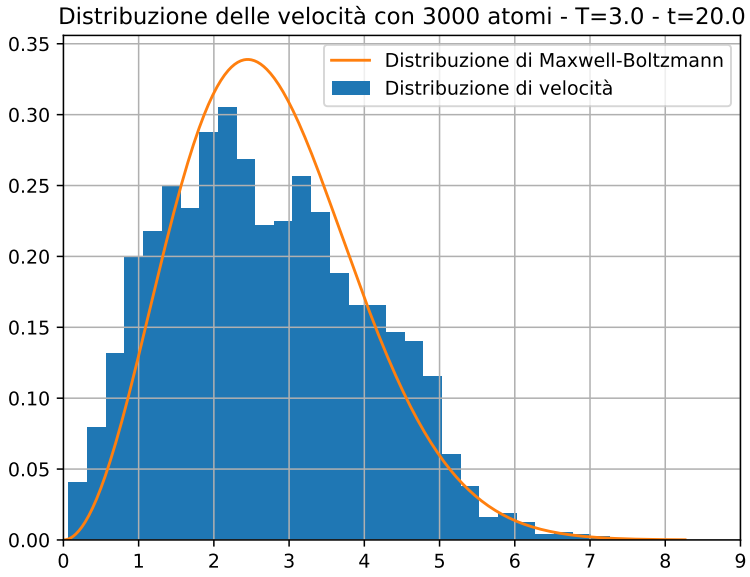


Figura 21: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=20.0.

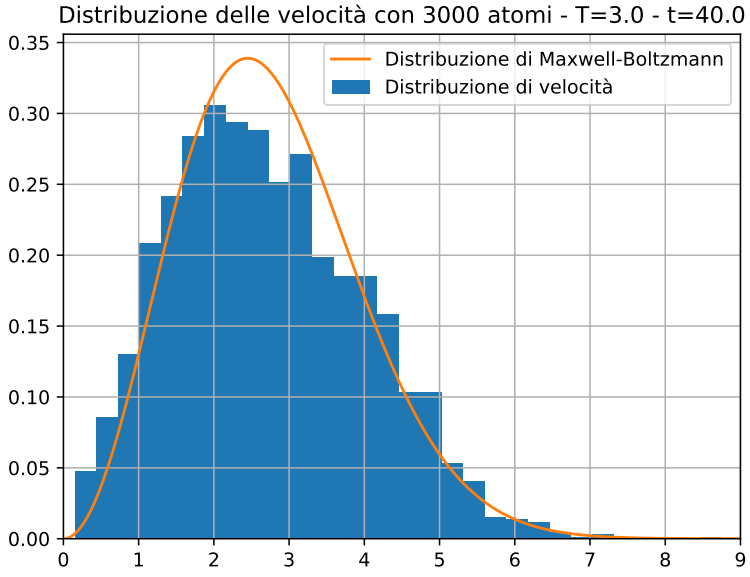


Figura 22: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=40.0.

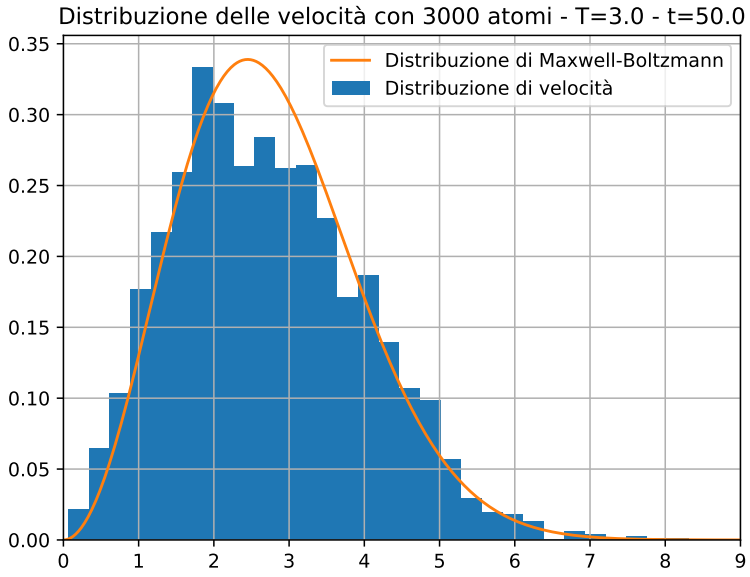


Figura 23: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=50.0.

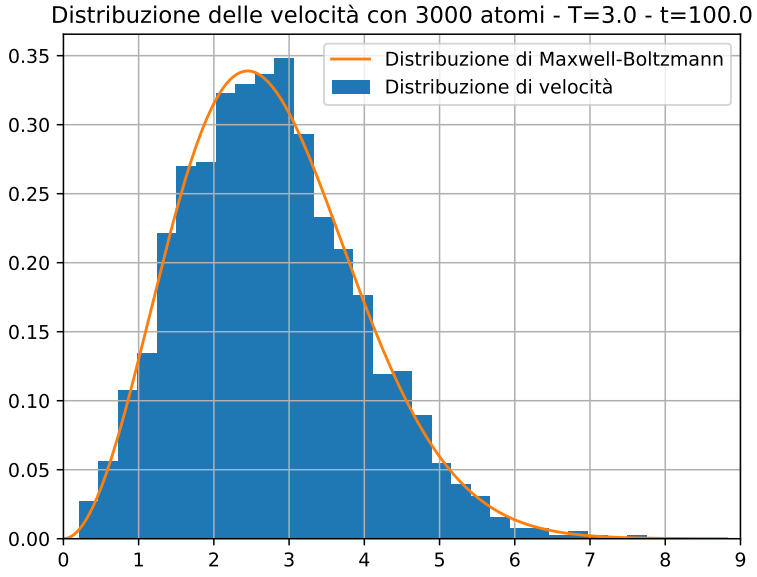


Figura 24: Distribuzione di velocità degli atomi di un sistema con 3000 atomi a t=100.0.

buzione di Maxwell-Boltzmann. Nelle varie figure si mostra la distribuzione a diversi tempi. Nella figura 24 si nota che la distribuzione segue perfettamente la distriuzione di Maxwell-Boltzmann.

4.6 Autocorrelazione delle misure

Si studia ora l'autocorrelazione delle misure. Innanzitutto si suppone che due misure siano scorrelate dopo una collisione tra due particelle; di conseguenza si è scelto di valutare nel seguente modo il tempo tra due collisioni:

$$\tau = \frac{d_0}{v_0} \tag{20}$$

dove:

- $v_0 = \sqrt{2T}$ = stima velocità media della particella
- $d_0 = \frac{L^2}{4\pi\sigma^2 N^{2/3}}$ = stima della distanza percorsa tra due collisioni
- T = temperatura
- L = lato del cubo
- N = numero di particelle

Arbitrariamente si è scelto di imporre $\tau = 2$ se il tempo tra due collisioni è < 2 unità di tempo: a volumi piccoli del sistema e temperature alte, ho pensato di mantenere $\tau = 2$ aiutasse a mantenere le misure del sistema scorrelate, senza questa regola, ci sono casi in cui si effettuano due misure a pochissimi step di integrazione di distanza.

L'autocorrelazione si calcola come:

$$\Gamma_X(t) = \frac{1}{N-t}\sum_{i=1}^{N-t}(X_i - \bar{X})(X_{i+t} - \bar{X}) \quad , \quad \bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{21}$$

Si guardano ora i grafici delle con le autocorrelazioni della temperatura sotto varie casistiche, si sono anche riutilizzati gli stessi dati partendo da istanti di tempo diversi per valutare la stessa autocorrelazione.

Si nota innanzitutto dai vari grafici che l'autocorrelazione della temperatura dipende dall'istante in cui si fa l'ultima misura: se la prima misura fosse stata fatta a $t = 0$ (primo grafico), la misura successiva avverrebbe in un istante in cui la correlazione è ancora alta; mentre se la prima misura fosse stata fatta a $t = 40$ o $t = 50$, la misura successiva sarebbe abbastanza scorrelata. Questo ci dice che misure intervallate da un tempo τ saranno scorrelate se prima si lascia evolvere il sistema per un po' di tempo. Ripensando allo studio della distribuzione di velocità, a $t = 0$ la distribuzione di velocità è una distribuzione uniforme (figura 15), mentre a $t = 40$ (figura 22) e $t = 50$ (figura 23) si è più vicini all'equilibrio. Pertanto può essere che la correlazione con misure distanziate di τ a $t = 0$ sia può alta per via del fatto che la distribuzione di velocità degli atomi è uniforme, mentre la correlazione negli altri due casi sia minore dato che si è più vicini all'equilibrio, ovvero la distribuzione di velocità è più simile ad una distribuzione di Maxwell-Boltzmann.

Alla luce di quest'ultima considerazione, nell'ottenere le curve di Van Der Waals si effettueranno misure distanziate di τ. Inoltre questa anomalia sulla correlazione è presente solo per le misure di temperatura; infatti, per la pressione, le misure si scorrelano quasi istantaneamente, quindi è anche possibile iniziare le misure da subito, senza aspettare la fine del transitorio.

La modalità di effettuazione delle misure in un run è il seguente:

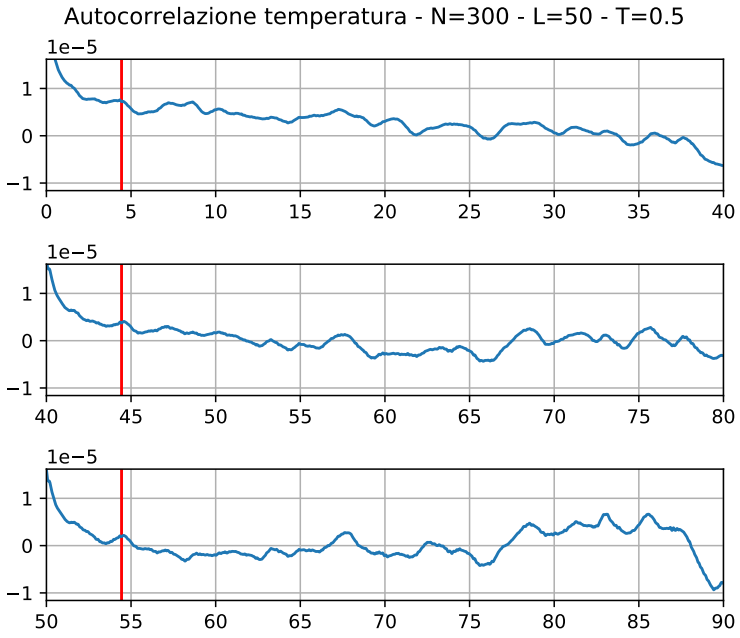


Figura 25: Autocorrelazione della temperatura del run con 300 atomi, 50 unità di lunghezza di lato, 0.5 di temperatura. La linea verticale rossa è l'istante della misura.

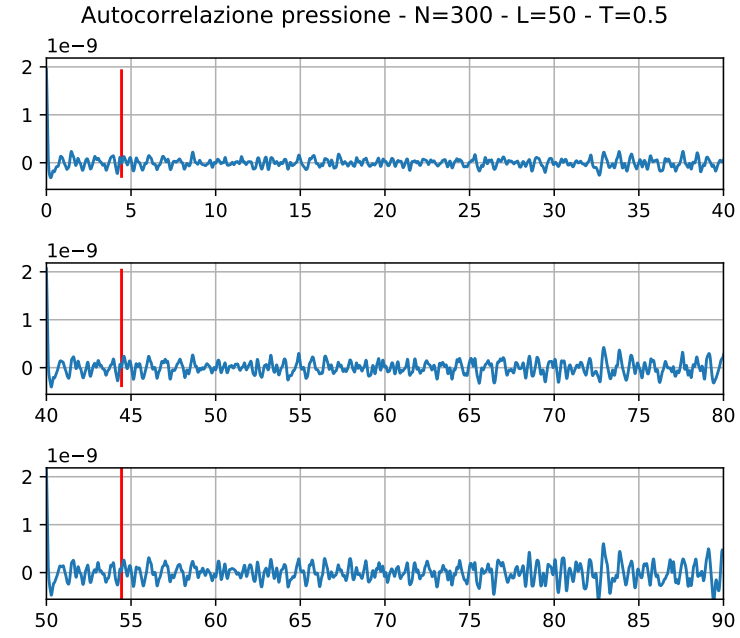


Figura 26: Autocorrelazione della pressione del run con 300 atomi, 50 unità di lunghezza di lato, 0.5 di temperatura. La linea verticale rossa è l'istante della misura.

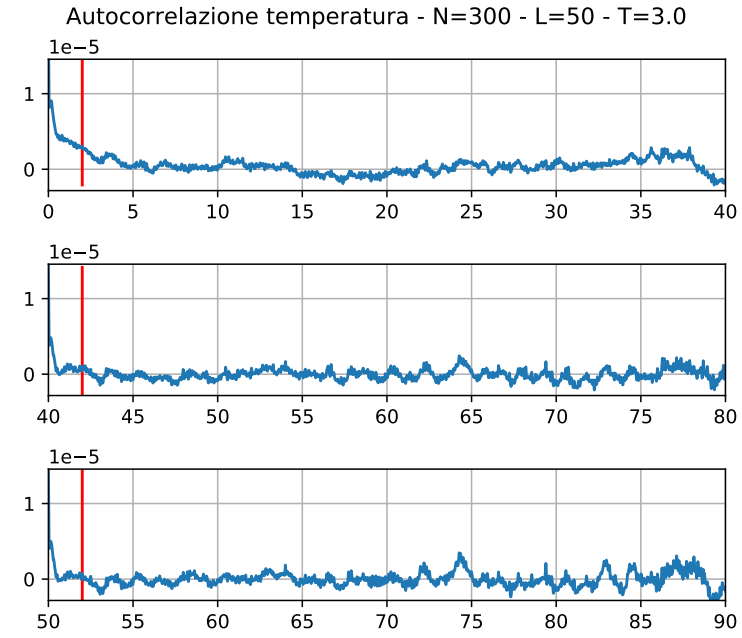


Figura 27: Autocorrelazione della temperatura del run con 300 atomi, 50 unità di lunghezza di lato, 3.0 di temperatura. La linea verticale rossa è l'istante della misura.

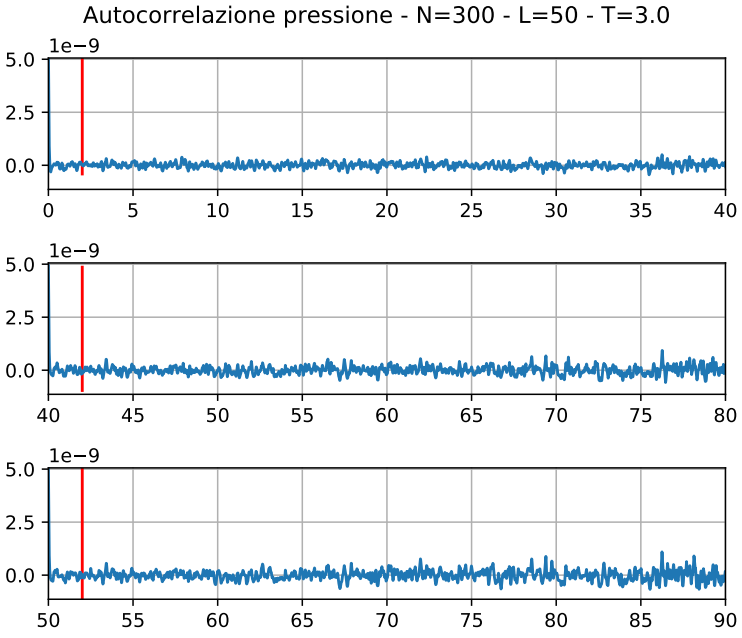


Figura 28: Autocorrelazione della pressione del run con 300 atomi, 50 unità di lunghezza di lato, 3.0 di temperatura. La linea verticale rossa è l'istante della misura.

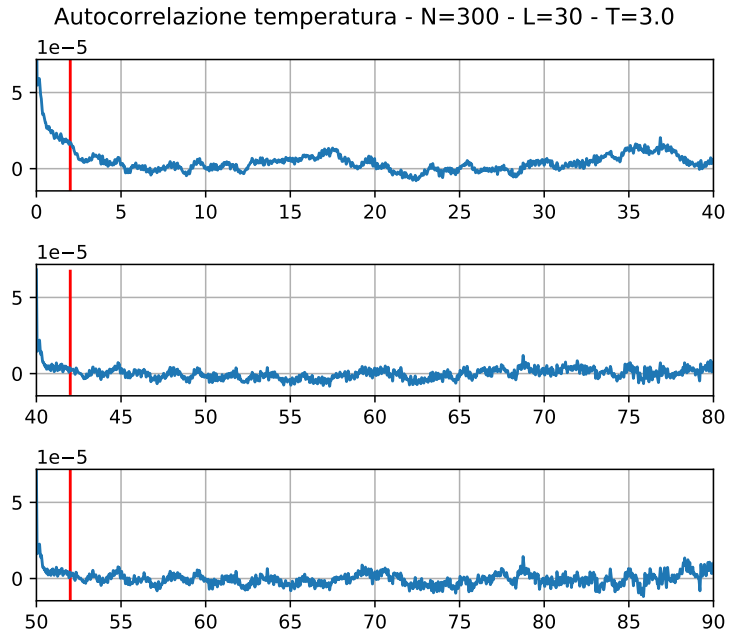


Figura 29: Autocorrelazione della temperatura del run con 300 atomi, 30 unità di lunghezza di lato, 3.0 di temperatura. La linea verticale rossa è l'istante della misura.

1. Tramite il tempo τ e la durata temporale di ciascuno step di integrazione, si calcola ogni quanti step di integrazione effettuare la misura. Si utilizza quindi il sottoprogramma `evolve` per far evolvere il sistema di questo numero di step (`evolve` fa evolvere il sistema di un numero di step scelto dalla funzione chiamante e restituisce misure di energia totale, energia potenziale, temperatura e pressione).

2. In `evolve`, ad ogni step di integrazione si memorizzano energia totale, energia potenziale, pressione e temperatura in quattro array differenti di dimensione pari al numero di step da effettuare (`arrayEnergy`, `arrayPotEnergy`, `arrayTemperature` e `arrayPressure`). Alla fine dell'evoluzione, prima di ritornare al sottoprogramma chiamante, si effettuano le medie dei valori memorizzati in questi array: questa è una singola misura.

3. Si scarta la prima misura dato che il sistema sicuramente non si trovava all'equilibrio. Inoltre si riscalano le velocità delle particelle per ottenere la temperatura desiderata impostata in `requiredTemp`.

4. Se il valore della misura della temperatura si discosta troppo dalla temperatura desiderata, si scartano tutte le misure effettuate finora e si riscalano le velocità per ottenere la temperatura desiderata. Ho scelto di utilizzare questo criterio dato che, se la temperatura misurata è fuori dal limite, quasi sicuramente si giunge all'equilibrio in una temperatura diversa da quella desiderata. Questo accorgimento fa risparmiare risorse del calcolatore.

5. Si effettuano 60 misure accettate in questo modo.

6. Si dividono le misure in due gruppi e si fa la media: le prime 30 misure daranno $\bar{T}_1$ e le 30 misure successive daranno $\bar{T}_2$.

7. Se $\bar{T}_1$ e $\bar{T}_2$ sono uguali entro un certo errore, si è giunti all'equilibrio.

   - Se si è all'equilibrio e $\bar{T} = \frac{\bar{T}_1+\bar{T}_2)}{2}$ è pari alla temperatura desiderata entro un certo errore ε_{req}, si salva il run (si è chiesto $\varepsilon_{req} = 1\%$).
   - Altrimenti se solo $\bar{T}_2$ è pari alla temperatura desiderata entro un certo errore, si scartano le prime 30 misure e ne vengono effettuate altre 30. Si torna così al punto 6.
   - Se invece anche $\bar{T}_2$ è diversa dalla temperatura desiderata entro l'errore, si scartano le prime 30 misure, si riscalano le velocità per ottenere la temperatura desiderata, si rieffettuano 30 misure e si torna al punto 6.

L'errore rispetto alla temperatura desiderata è valutata nel modo seguente:

$$\varepsilon_{mes} = \left| \frac{T_{req} - T_{mes}}{T_{req}} \right| \tag{22}$$

Nel programma principale si richiede una certa accuratezza della misura, ovvero si fissa un errore richiesto massimo detto ε_{req}: minore è ε_{req}, più la temperatura misurata deve essere vicina alla temperatura richiesta. Si applica il criterio del punto 4 se $\varepsilon_{mes} > 5\varepsilon_{req}$.

Nel salvataggio del run, si salvano: tempo del run in unità naturali, dimensione della scatola, energia totale, temperatura, pressione e le rispettive deviazioni standard.

4.7 Isoterme

Dopo aver definito le modalità di misura, simulano le isoterme. Si definisce il volume specifico in scala logaritmica come:

$$x = \log_{10} \frac{V}{N} \tag{23}$$

Con abuso di linguaggio, chiamerò x volume dato che il volume vero e proprio non viene mai utilizzato. Si parte da un volume iniziale $x_0 = 3.0$ e si arriva a $x_f = -0.1$ con $\Delta x = -0.1$. Sono state fatte simulazioni con diverso numero di atomi ($N = 300, 500, 1000$) e diverse temperature.

4.7.1 Considerazioni generali

Per motivi successivamente spiegati, si utilizza la terminologia "alte temperature" per temperature $T \geq 0.8$ gradi, mentre si riserva la locuzione "basse temperature" per $T \leq 0.8$ gradi, anche se la temperatura $T = 0.8$ è una bassa temperatura (avviene una transizione di fase).

Si nota in tutti i casi studiati un iniziale andamento approssimabile ad un gas perfetto (linea gialla sottile), ma si ha poi una successiva deviazione. A temperature più basse si ha una deviazione che avviene prima rispetto alle temperature più alte.

Nella figura 32 si sono sovrapposte le isoterme per $T \geq 1.0$ gradi ottenute con un diverso numero di atomi: si nota che l'andamento è molto simile, quindi non ci sono rilevanti effetti di volume finito o ridotto numero di particelle. Discorso invece diverso per le isoterme a $T = 0.8$ (figura 34): a questa temperatura si notano degli andamenti non esattamente sovrapponibili intorno al punto di flesso; in tal caso gli effetti di limitato numero di particelle e volume finito sono maggiori.

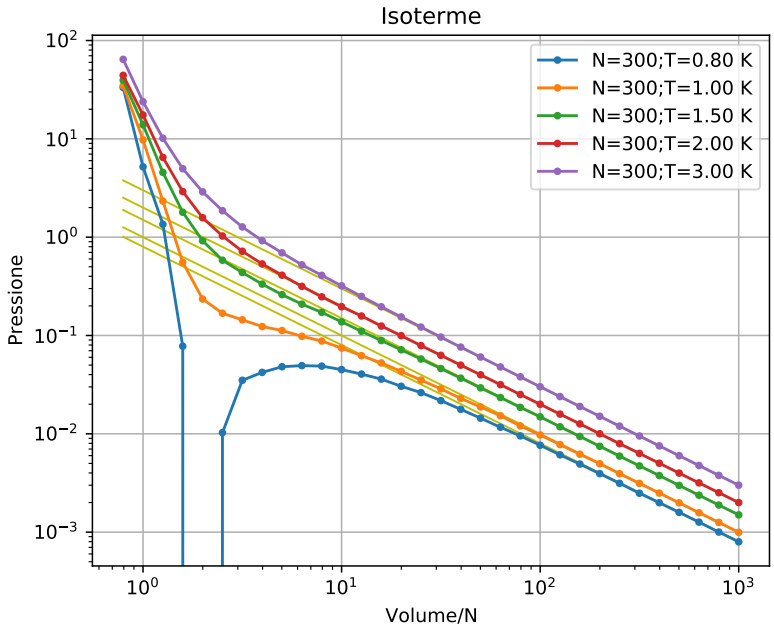


Figura 30: Isoterme ad alte temperature con 300 atomi.

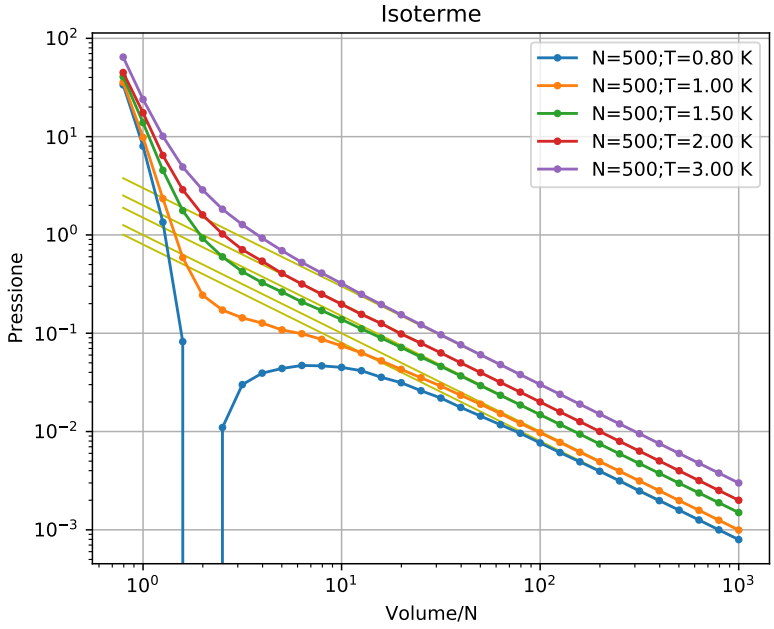


Figura 31: Isoterme ad alte temperature con 500 atomi.

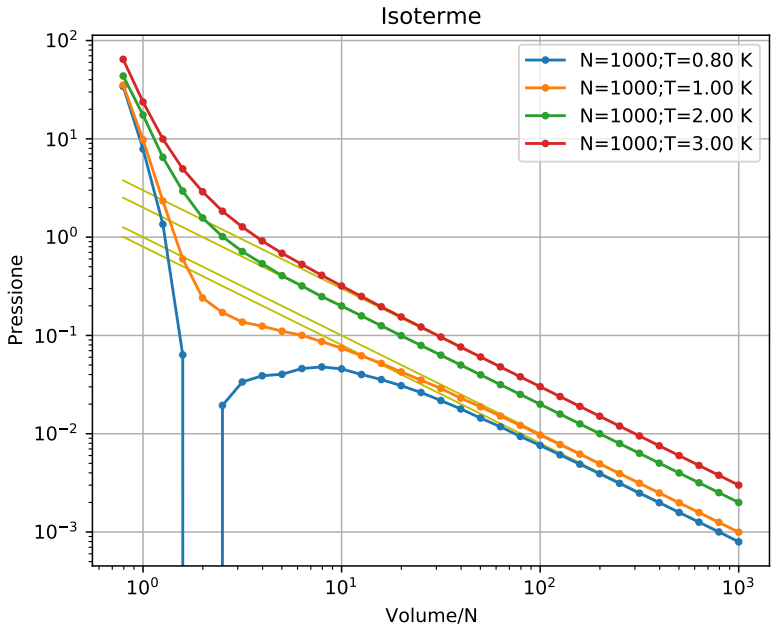


Figura 32: Isoterme ad alte temperature con 1000 atomi.

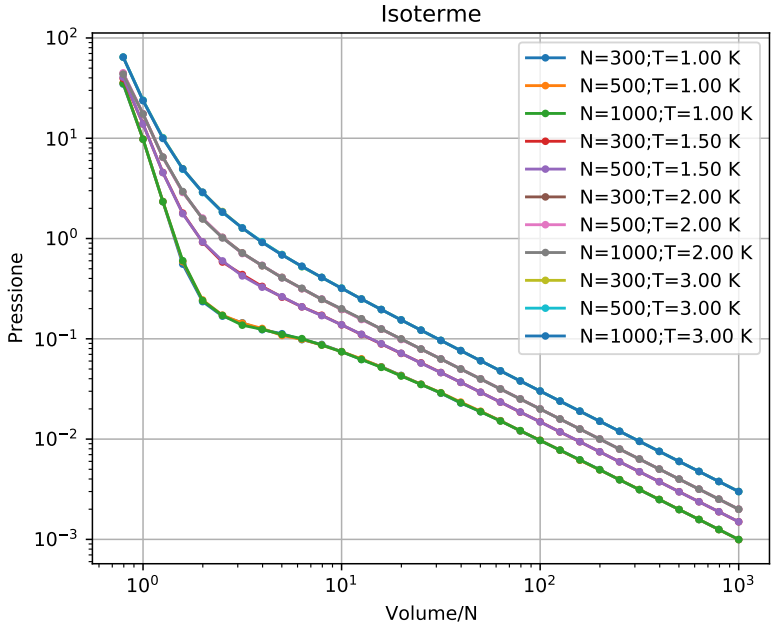


Figura 33: Isoterme isoterme sovrapposte con 300, 500, 1000 atomi.

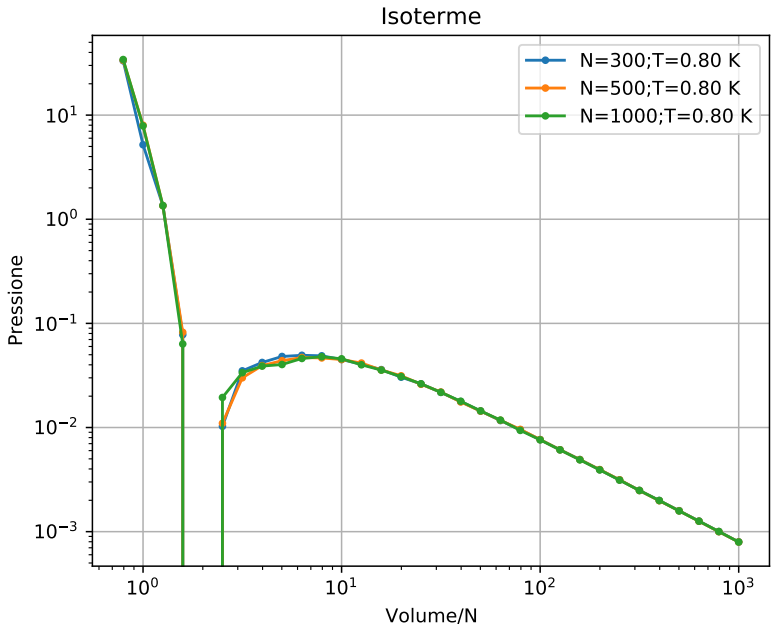


Figura 34: Isoterme isoterme sovrapposte a $T = 0.8$ con 300, 500, 1000 atomi.

4.7.2 Gas reale

Dopo aver ottenuto le curve, si è provato ad eseguire un fit con l'equazione di stato dei gas reali. La legge di Van Der Waals con unità nel SI è la seguente:

$$\left(P + \frac{a}{V_m^2}\right)(V_m - b) = k_B N_A T \tag{24}$$

dove:

- $V_m = \frac{N_A V}{N}$ = volume molare
- N_A = numero di Avogadro

Per l'argon si ha:

- $a = 0.1363 \, \frac{\mathrm{J\,m^2}}{\mathrm{mol^2}}$
- $b = 0.00003219 \, \frac{\mathrm{m^3}}{\mathrm{mol}}$

Con opportuni passaggi matematici, si giunge alla seguente espressione in unità ridotte per la pressione:

$$P = \frac{T}{\bar{V} - \bar{b}} - \frac{\bar{a}}{\bar{V}^2} \tag{25}$$

dove $\bar{V} = \frac{V}{N}$. Pertanto, in unità naturali si ha:

- $\bar{a} = 5.77$

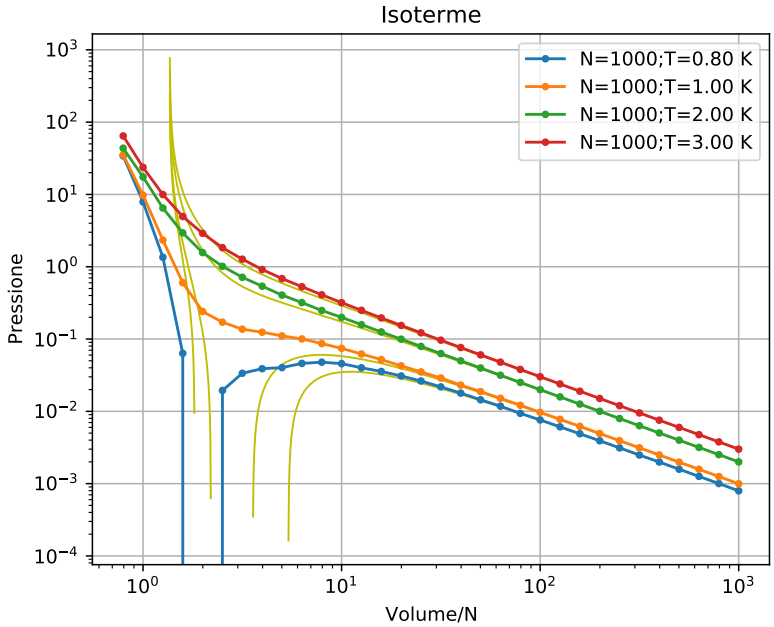


Figura 35: Le linee spesse sono le isoterme ad alta temperatura simulate. Le isoterme gialle sottili sono le isoterme ottenute con l'equazione di Van Der Waals (equazione (25)) con i parametri dell'argon. Si mostrano solo i grafici con $N = 1000$ dato che in precedenza si è detto che le curve si sovrappongono abbastanza bene a tutti gli N.

Temperatura	a	b
$T = 3.0$	1.52	1.01
$T = 2.0$	2.94	1.21
$T = 1.0$	4.24	1.43
$T = 0.8$	5.31	1.84

Tabella 1: Valori di a e b ottenuti tramite fit per le curve ottenute simulando un sistema di $N = 1000$ atomi. Il fit è ottenuto considerando i 25 punti associati a volume grande (i 25 punti più a destra).

- $\bar{b} = 1.36$

Si mostrano nella figura 35 i grafici con le isoterme ottenute dall'equazione (25). Si nota che le curve simulate seguono l'andamento predetto dall'equazione solamente a grandi volumi, ovvero solamente nel limite di gas perfetto. Non ho trovato modo per fittare i dati in modo che venissero gli stessi coefficienti a e b per tutte le curve. Nella tabella 1 si mostrano alcuni valori ottenuti tramite fit: si nota che sono valori tra loro molto discordanti e abbastanza lontani dal valore tabulato per l'argon. Inoltre, il valore ricavato dal fit è molto dipendente dal numero di punti considerati, effettuare un fit con 25 punti (come in tabella) o con 24 risulta in valori di a e b molto diversi. Provando anche a fittare una sola curva e disegnare isoterme a temperatura diversa con gli stessi parametri, si nota che l'andamento analitico è molto diverso dalle isoterme delle simulazioni.

4.7.3 Temperatura critica

La temperatura critica è la temperatura oltre la quale non si può più avere condensazione di un gas in liquido. Si è detto che, indipendentemente dal numero di atomi, le isoterme si sovrappongono sufficientemente bene, come mostrato nella figura 33 e nella figura 34. Dalla figura 30 si evince che la temperatura critica è compresa tra $0.80 < T < 1.0$, pertanto si sono tracciate alcune isoterme tra queste temperature. I risultati si trovano nella figura 36.

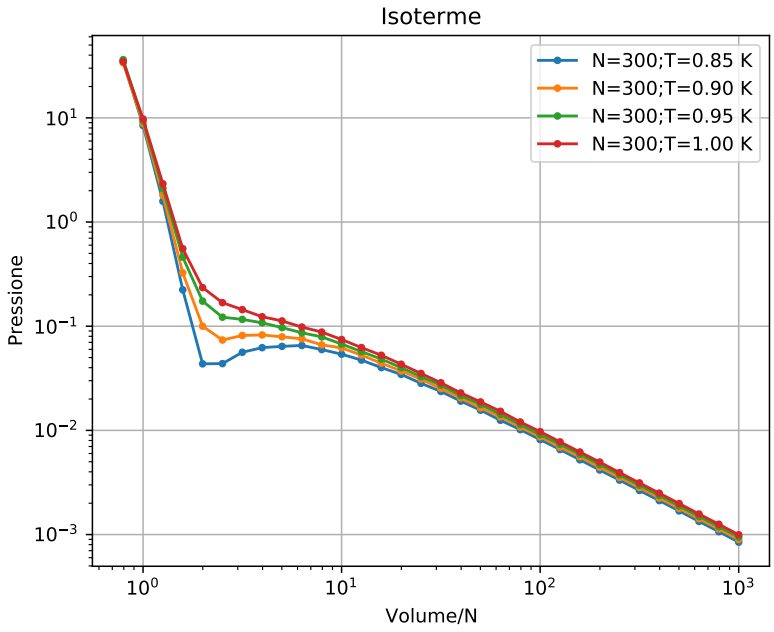


Figura 36: Isoterme con 300 atomi per trovare la temperatura critica. Si nota che la temperatura critica è circa $T = 0.95$.

Dalla figura 36 si nota che la temperatura critica del sistema è $T = 0.95$; nel Sistema Internazionale la temperatura critica del sistema sarebbe circa $T = 114\,\mathrm{K}$. La temperatura critica dell'argon è invece $T = 151\,\mathrm{K}$.

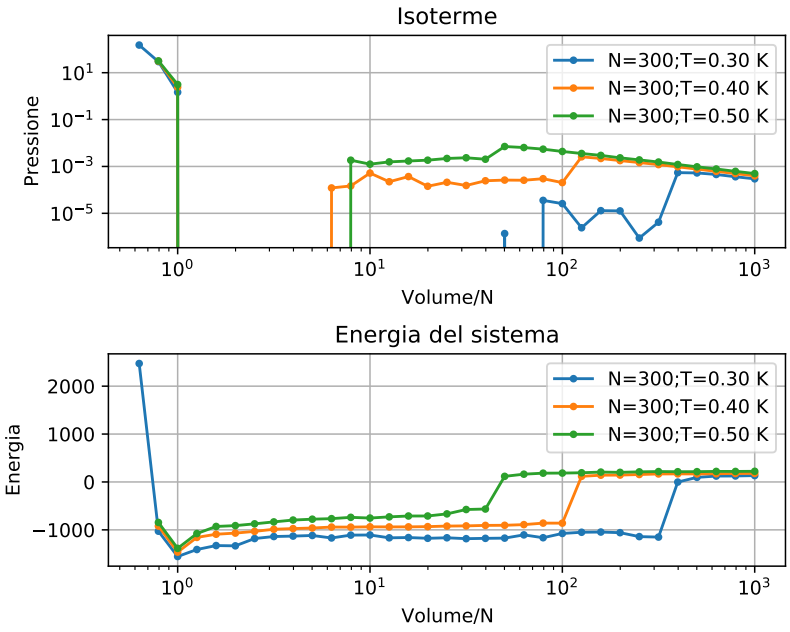


Figura 37: Isoterme ed energie con 300 atomi a bassa temperatura. Si nota che ad un certo punto sulle isoterme si ha un andamento piatto: questa è conseguenza di una transizione di fase in corso, sono presenti goccioline di liquido che condensano. L'andamento a $T = 0.3$ è molto incerto e poco piatto: questo è spiegabile col fatto che il grafico è in scala logaritmica e il valore della pressione è basso con un errore sulle misure dell'ordine della misura stessa. Guardando il grafico con le energie non si hanno più dubbi: si ha una transizione di fase, il salto in energia è l'energia latente di condensazione e indica la formazione di gocce

4.7.4 Transizione di fase

A basse temperature invece, non si è studiato il valore numerico dei coefficienti a e b, ma si è notato un fatto inaspettato dato il limitato numero di atomi coinvolti: la formazione di gocce.

In figura 37 si hanno sia le isoterme, sia le energie. Sulle isoterme si nota che ad un certo punto la curva assume comportamento piatto, compatibile con una fase liquido-gas. Ponendo invece le energie sull'asse delle ascisse, si nota nel grafico sotto che si hanno effettivamente delle transizioni di fase. Si nota che il volume a cui avviene la transizione di fase varia a seconda del volume: temperature maggiori richiedono un volume più piccolo per far avvenire la transizione. Lo stesso si osserva anche con 500 e 1000 atomi.

4.8 Fusione di un cristallo

L'esercizio successivo è lo studio della temperatura di fusione di un cristallo. Si parte da un cristallo stabile, ovvero si parte da un reticolo FCC e si trova il volume tale per cui il cristallo ha pressione nulla a temperatura zero (si trova il volume tale per cui $P(T = 0) = 0$). Per trovare tale volume, mi sono messo a $x_0 = 0$, con $x =$ volume specifico definito dall'equazione (23) e ho aggiunto un $\Delta x = 0.001$ alla volta finché non raggiungo pressione nulla. Con questo metodo non sono riuscito a trovare un volume tale per cui la pressione fosse nulla con un passo reticolare minore del cut off.

Tuttavia ho notato all'aumentare del volume, la pressione diminuiva ed era presente un valore tale per cui la pressione passava da positiva a negativa. Ho scelto come volume del cristallo l'ultimo volume tale per cui la pressione era positiva, è risultato che $x = 0.788$.

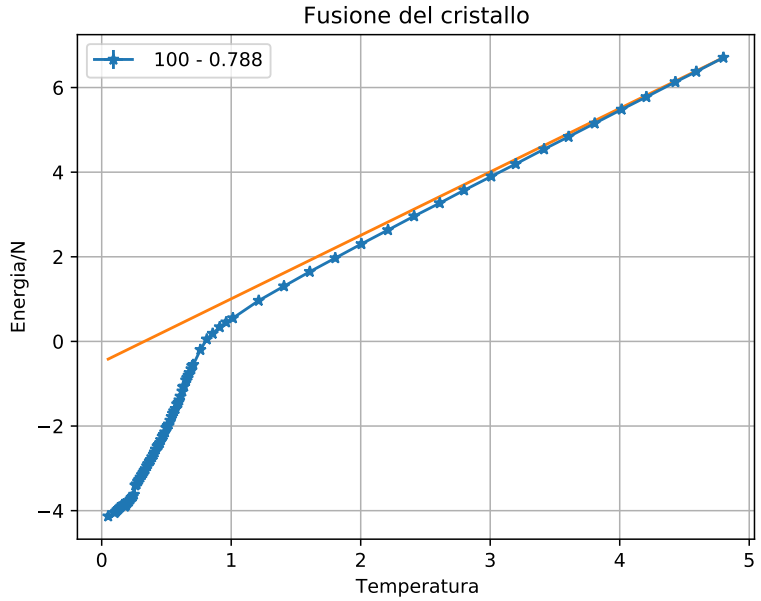


Figura 38: Fusione del cristallo, la linea arancione è una retta con pendenza $C_V^{gas} = \frac{3}{2}$

Facendo successivamente fondere il cristallo, si ottiene il grafico in figura 38. Si nota che a basse temperature si ha una piccolissima transizione di fase intorno a $T = 0.3$. Nella figura successiva (figura 39) si vede meglio che la transizione di fase si trova a circa $T = 0.26$. Si nota che la legge di Dulong-Petit è rispettata prima della transizione di fase, ma non successivamente. Tra $T = 0.30$ e $T = 1.0$ si ha un andamento molto strano, si suppone che sia un liquido che passa allo stato di gas senza transizione di fase, infatti a temperature $T > 1.0$ si ha il calore specifico di un gas perfetto.

Ho supposto che l'andamento tra $T = 0.30$ e $T = 1.0$ sia dovuto alla elevata densità del cristallo, ho effettuato quindi studi con volumi maggiori: in particolare con un cristallo di volume $x = 1.551$ (passo reticolare pari al cut off).

Nella figura 40 si nota un andamento molto in accordo con il calore specifico di un gas perfetto, si osserva successivamente anche una transizione di fase liquido-gas. Tuttavia, ingrandendo a basse temperature, nella figura 41 si nota che l'andamento segue sì la legge di Dulong Petit, ma non si parte da uno stato di energia minima, infatti le misure a $T < 0.19$ vengono fatte in un minimo locale del potenziale del cristallo, mentre a $T = 0.19$ l'energia improvvisamente crolla: questo è dovuto al fatto che, aumentando la temperatura, il sistema può esplorare regioni dello spazio delle fasi prima precluse. Il sistema può quindi trovare dei minimi del potenziale più profondi. Una seconda interpretazione è che avviene una transizione di fase solido-solido: gli atomi si ridispongono in una configurazione energeticamente più vantaggiosa che non era possibile ottenere in precedenza a causa della presenza di una barriera di potenziale. Si è notato che la legge di Dulong-Petit viene comunque rispettata in entrambi i casi, quindi in entrambi i casi si ha un solido classico.

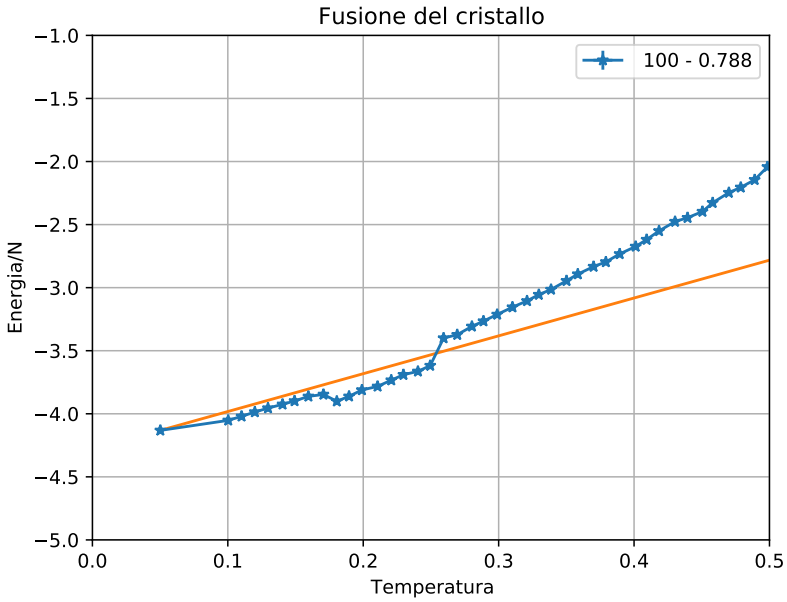


Figura 39: Fusione del cristallo, si è ingrandito nell'intervallo $[0, 0.5]$. La linea arancione segue la legge di Dulong Petit.

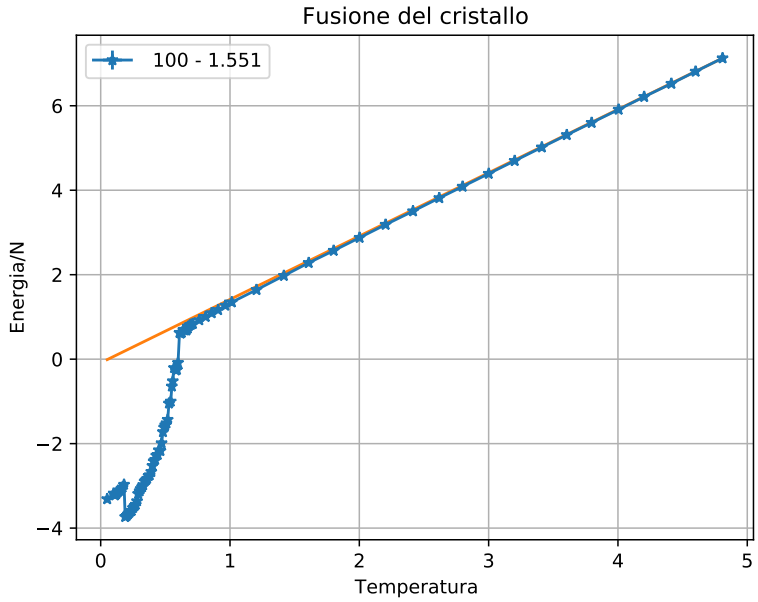


Figura 40: Fusione del cristallo, la linea arancione è una retta con pendenza $C_V^{gas} = \frac{3}{2}$

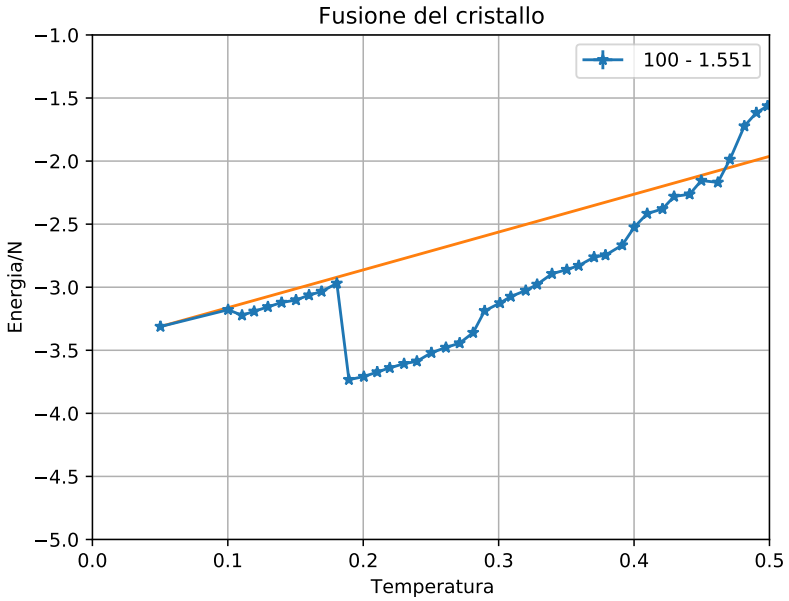


Figura 41: Fusione del cristallo, si è ingrandito nell'intervallo $[0, 0.5]$. La linea arancione segue la legge di Dulong Petit.

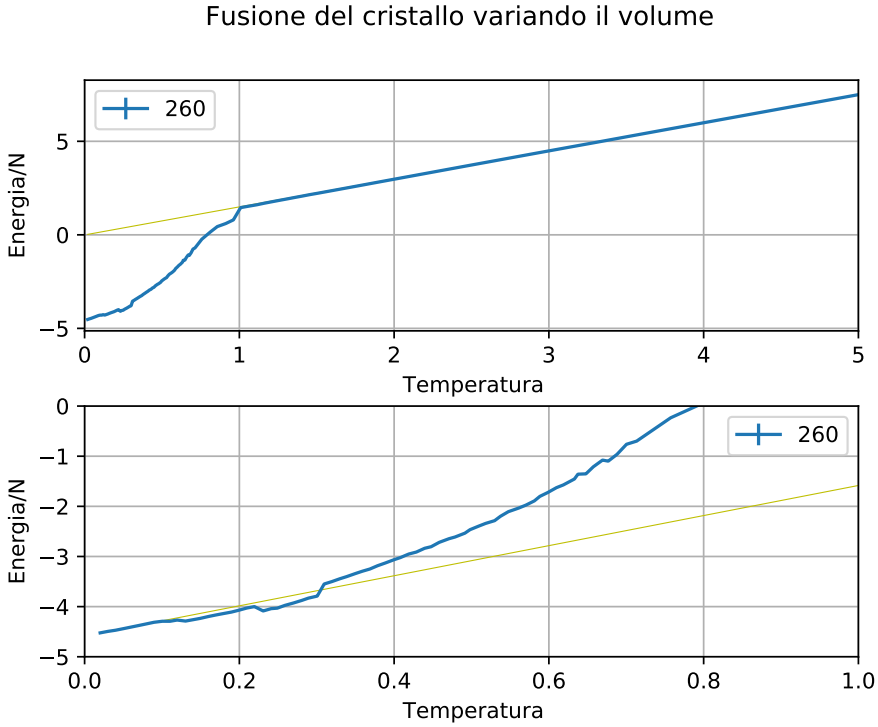


Figura 42: Fusione del cristallo variando il volume per avere pressione nulla (o sotto soglia). Il ΔT tra un run e l'altro è lo stesso di tutte le altre simulazioni di fusione.

Ho provato a fare uno studio utilizzando 260 atomi variando il volume prima di ogni run per annullare la pressione. Nella figura 42 si mostrano i risultati. Si nota che a circa $T < 0.3$ il calore specifico segue la legge di Dulong-Petit; successivamente inizia ad esserci una deviazione dalla legge (analogo al caso a volume fissato), infine a circa $T = 1.0$ si nota una transizione di fase. A $T > 1.0$ si ha il calore specifico di un gas perfetto a volume costante, questo è un andamento inaspettato dato che il volume non è costante, mentre la pressione lo è (prima di ciascun run si annulla la pressione): la regione che si comporta da gas perfetto dovrebbe avere pendenza $C_P = C_V + 1 = \frac{5}{2}$.

4.9 Condensazione del gas

Per uno studio più approfondito e per evitare di esplorare minimi locali nello spazio delle fasi (ovvero per evitare di esplorare cristalli che non si trovano nel minimo), ho fatto anche dei run di condensazione, ovvero sono partito dalla temperatura di $T = 5.0$ e sono arrivato ad una temperatura di $T = 0.01$. Questo è un utile esercizio dato che permette al sistema di posizionarsi nel minimo globale del potenziale quando si giungono a temperature molto basse. Ovviamente, affinché l'esercizio sia utile, non devo reinizializzare la posizione degli atomi per ciascun punto ottenuto, ovvero non devo riporre tutti gli atomi su un reticolo FCC, ma devo partire dalla posizione degli atomi alla fine del run precedente.

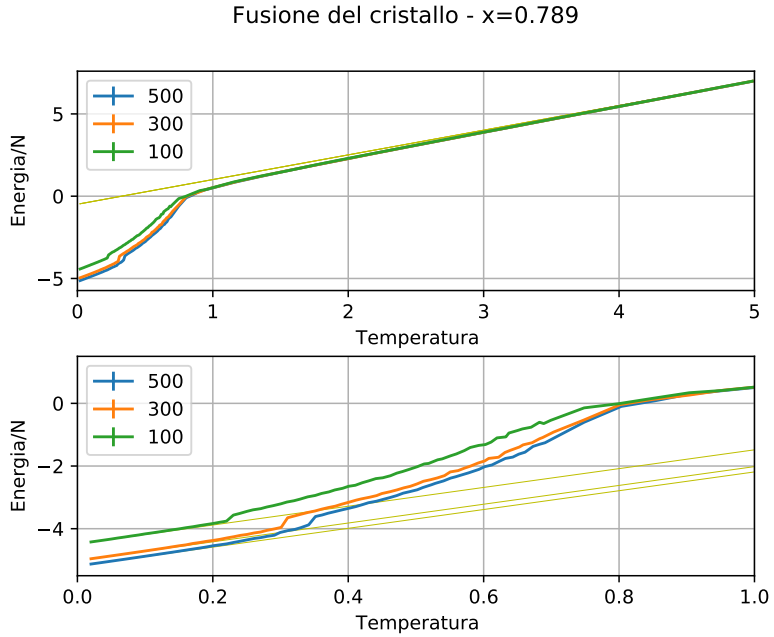


Figura 43: Simulazione di condensazione prima e solidificazione poi di un gas in un volume specifico $x = 0.789$

Per un gas perfetto monoatomico, il calore specifico a volume costante in unità ridotte è:

$$C_V = \frac{3}{2} \tag{26}$$

Nelle varie figure, la linea gialla del grafico in alto è una retta con pendenza C_V, quindi se le simulazioni sono compatibili con la linea gialla, si è in presenza di un gas perfetto monoatomico.

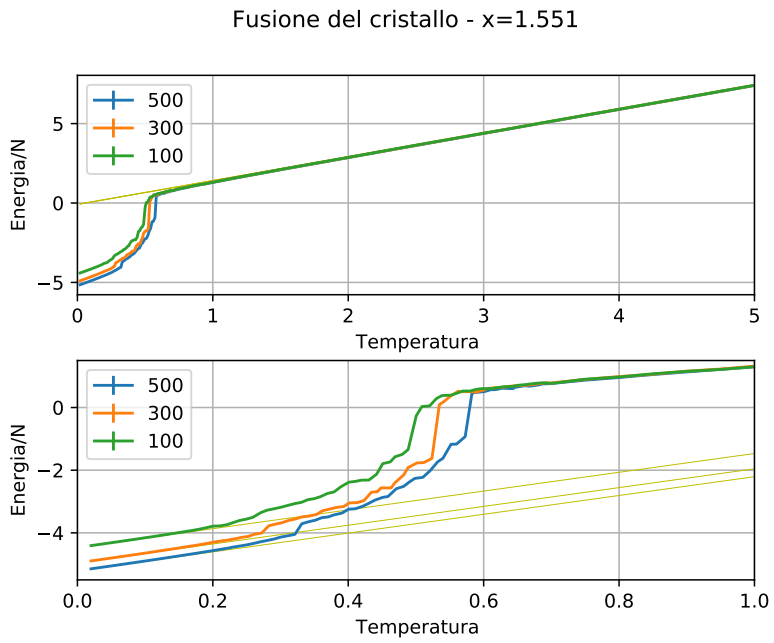


Figura 44: Simulazione di condensazione prima e solidificazione poi di un gas in un volume specifico $x = 1.551$

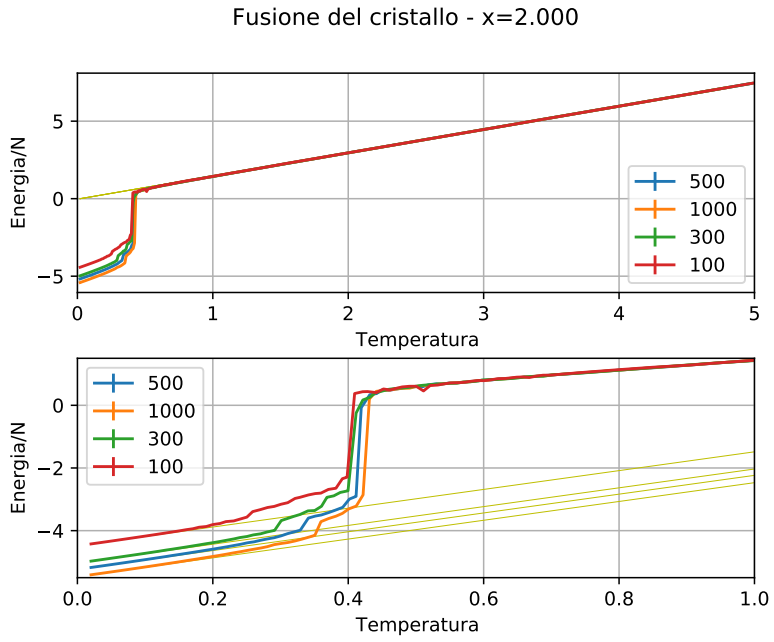


Figura 45: Simulazione di condensazione prima e solidificazione poi di un gas in un volume specifico $x = 2.0$

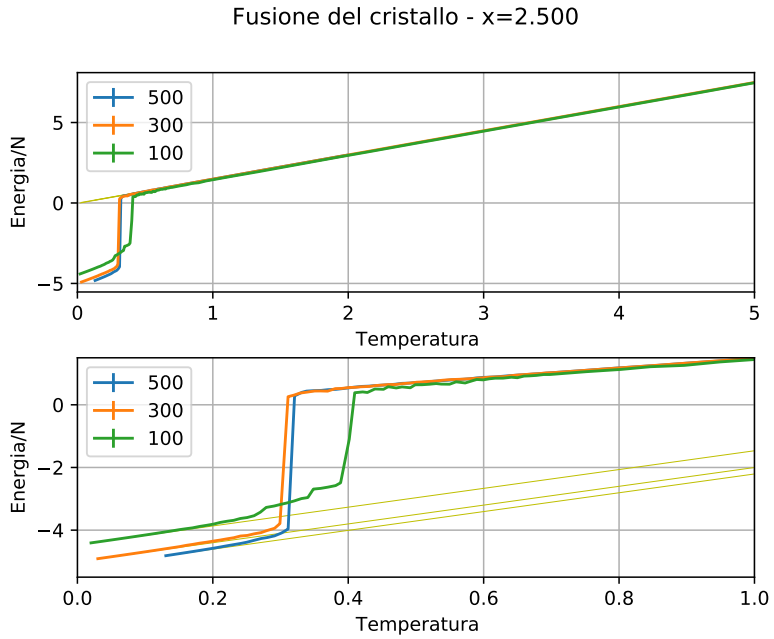


Figura 46: Simulazione di condensazione prima e solidificazione poi di un gas in un volume specifico $x = 2.5$

Si discute ora il caso con volume $x = 0.789$. Dal grafico si nota che a temperature molto alte ($T > 2.0$), il calore specifico segue proprio l'andamento di un gas perfetto; a temperature tra $0.8 < T < 2.0$ si ha una iniziale deviazione, ma si può ancora considerare gas perfetto; a temperature $T < 0.8$ il sistema smette di essere un gas perfetto, la pendenza diventa maggiore; vista l'assenza di una transizione di fase, è ragionevole supporre che si ha ancora uno stato di gas (classico) ultra compresso. Chiamiamo *temperatura limite* T_L la temperatura intorno a $T = 0.8$, quindi a $T > T_L$ si ha il gas perfetto. Diminuendo ulteriormente la temperatura, a seconda del numero di atomi, si nota un salto in energia: questa è una transizione di fase gas-solido; a temperature minori della temperatura di sublimazione si nota che l'andamento segue tanto meglio la legge di Dulong Petit quanto la temperatura è più vicina allo zero.

Per i volumi $x = 1.551$ e $x = 2.0$ invece, a temperature più alte si ha un ottimo accordo con l'andamento di un gas perfetto fino alla transizione di fase; questa transizione è abbastanza netta per $x = 1.551$, mentre è molto netta per $x = 2.0$. Questo suggerisce che l'energia latente di vaporizzazione sia maggiore se si ha un volume maggiore a disposizione. Successivamente alla transizione di fase, si ha una fase interpretabile come fase liquida. Diminuendo ulteriormente la temperatura si ha una seconda transizione di fase, in questo caso è ragionevole interpretare il salto di energia come transizione di fase liquido-solido. A conferma di questa interpretazione, a temperature minori, l'andamento dell'energia segue bene la legge di Dulong-Petit.

Per il volume maggiore ($x = 2.5$) si ha anche in questo caso un andamento che segue il calore specifico di un gas perfetto fino alla transizione di fase. Questa transizione di fase però (eccetto nel caso di 100 atomi) è compatibile con una transizione di fase gas-solido. Dato che il volume è grande, la pressione è molto piccola: questo significa che a basse pressioni è possibile che un solido sublimi. A temperature minori si ha accordo con la legge di Dulong Petit. Per il caso con 100 atomi,

suppongo che il diverso andamento sia dato da effetti di ridotto numero di particelle.

4.10 Spostamento medio

Durante le simulazioni della condensazione di un gas, si è salvato anche lo spostamento medio calcolato come:

$$\Delta r(t) = \sqrt{\frac{1}{N}\sum_i [r_i(t) - r_i(0)]^2} \tag{27}$$

Più rigorosamente questa equazione sarebbe la radice quadrata dello spostamento quadratico medio, ma per abuso di linguaggio lo si chiama spostamento medio; lo spostamento medio propriamente detto è nullo dato che il centro di massa è fermo.

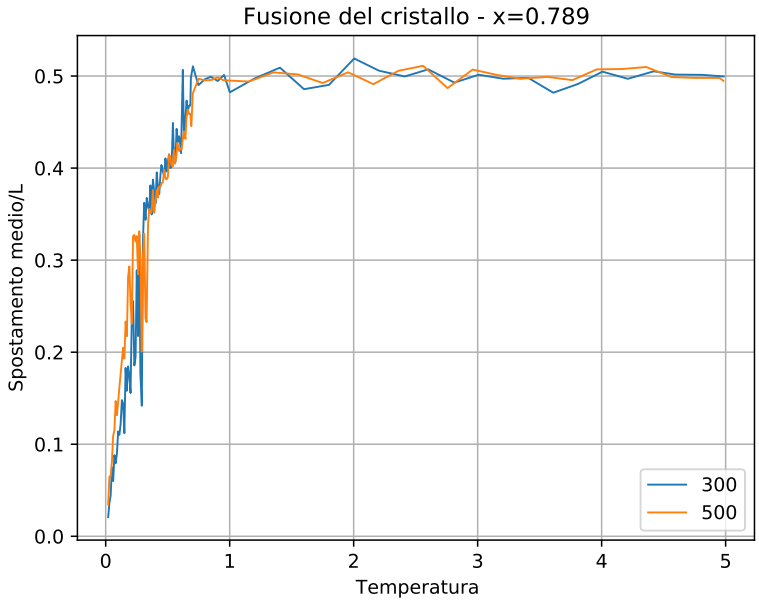


Figura 47: Spostamento medio degli atomi con volume $x = 0.789$.

Ho omesso i grafici con 100 atomi dato che a basse temperature, l'andamento è molto oscillante, questo è dovuto al fatto che si ha un limitato numero di atomi e le fluttuazioni sono molto importanti; con un numero maggiore di atomi le fluttuazioni vengono compensate e distribuite su più atomi. Inoltre i valori sull'asse delle ordinate sono sono lo spostamento medio diviso il lato del cubo. Essendo il cubo di periodicità L, ci si aspetta che $\Delta r(t) \simeq \frac{L}{2}$ per un gas perfetto.

Si nota che in figura 47 a $x = 0.789$ (si ricorda che vale l'approssimazione di gas perfetto a temperature $T > T_L$), lo spostamento medio è pari a $L/2$, questo vuol dire che gli atomi del sistema si muovono liberamente in tutto il volume. Tuttavia sotto la temperatura limite T_L, lo spostamento medio inizia a diminuire. Nella regione di validità della legge di Dulong-Petit si ha un andamento lineare.

A volumi maggiori invece, appena avvengono le trasizioni di fase gas-liquido, lo spostamento medio diminuisce molto velocemente (figure 48 e 49) e inizia a diminuire linearmente finché non si ha $\Delta r(t, T = 0) = 0$.

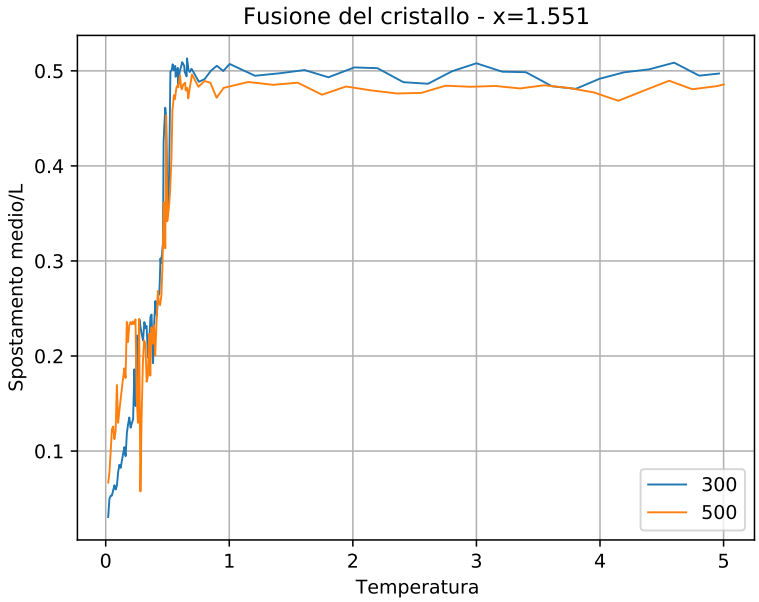


Figura 48: Spostamento medio degli atomi con volume $x = 1.551$.

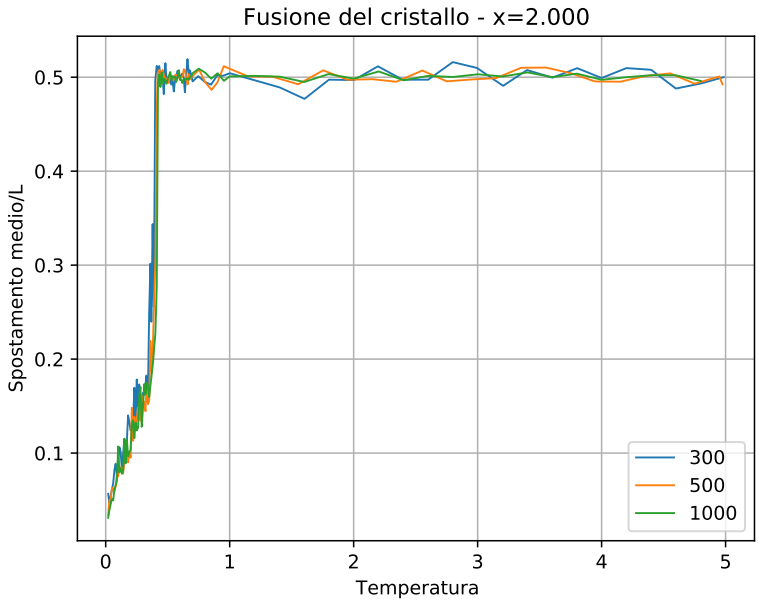


Figura 49: Spostamento medio degli atomi con volume $x = 2.0$.

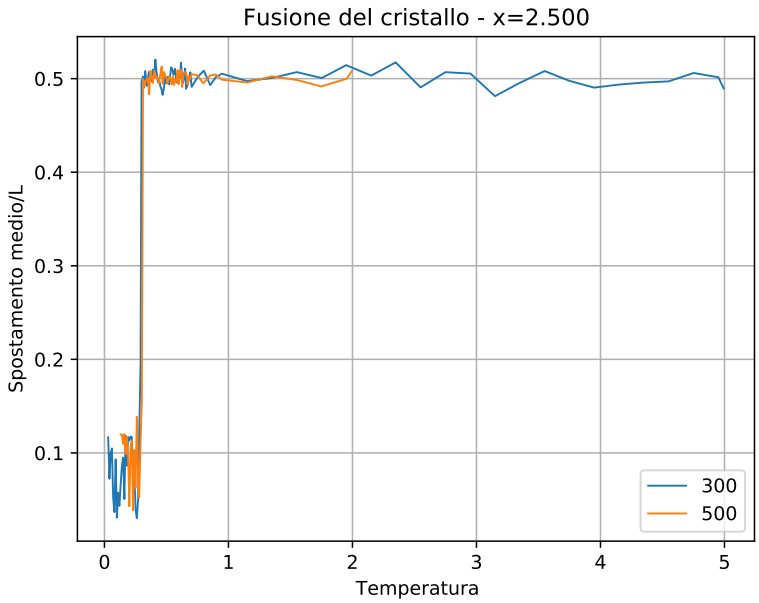


Figura 50: Spostamento medio degli atomi con volume $x = 2.5$.

A grande volume, in figura 50, si nota una anomalia: l'andamento di $\Delta r(t)$ diminuisce improvvisamente e rimane stabile su un valore singolo. Suppongo questo sia dovuto al fatto che il cristallo che si è formato ruota: infatti durante l'inizializzazione ci si è messi nel sistema del centro di massa (momento totale nullo), ma è ovviamente possibile che il momento angolare non sia nullo.

4.11 Distribuzione radiale

Per lo studio della distribuzione radiale, ci si aspetta che a basse temperature, il cristallo sia impacchettato nel suo volume, quindi tutte le distanze tra gli atomi sono piccole. Nel limite di un gas perfetto invece ci si aspetta che gli atomi occupino tutto il volume, pertanto la distribuzione radiale sarà più distribuita.

Durante la condensazione di un gas, al momento del salvataggio di un run, ho salvato tutte distanze tra gli atomi allocando un array di dimensione $N \times (N-1)$. Le distanze sono state poi salvate in un file, il quale è stato successivamente importato in Python per generare gli istogrammi, in particolare ho utilizzato 500 colonne.

Si è studiata la distribuzione radiale con $N = 200$ atomi in un volume $x = 2.0$. Guardando la figura 56 si nota che, indipendentemente dal numero di atomi, la temperatura di vaporizzazione è circa $T_V = 0.4$, pertanto si suppone che anche nel caso con $N = 200$ si abbia la stessa temperatura di vaporizzazione. Si guarda ora in figura 51 la distribuzione radiale a varie temperature.

Si nota che a $T < T_V$ si ha un cristallo formato, i picchi sono tutti intervallati alla stessa distanza, è quindi lecito supporre che i picchi si trovino in prossimità di multipli del passo reticolare e la posizione del primo picco è il passo reticolare.

A temperature intorno a T_V si ha un maggior numero di particelle diffuse nel volume (la distanza massima tra due particelle è $L\frac{\sqrt{3}}{2}$ dato che è metà diagonale del cubo), mentre il corpo principale è

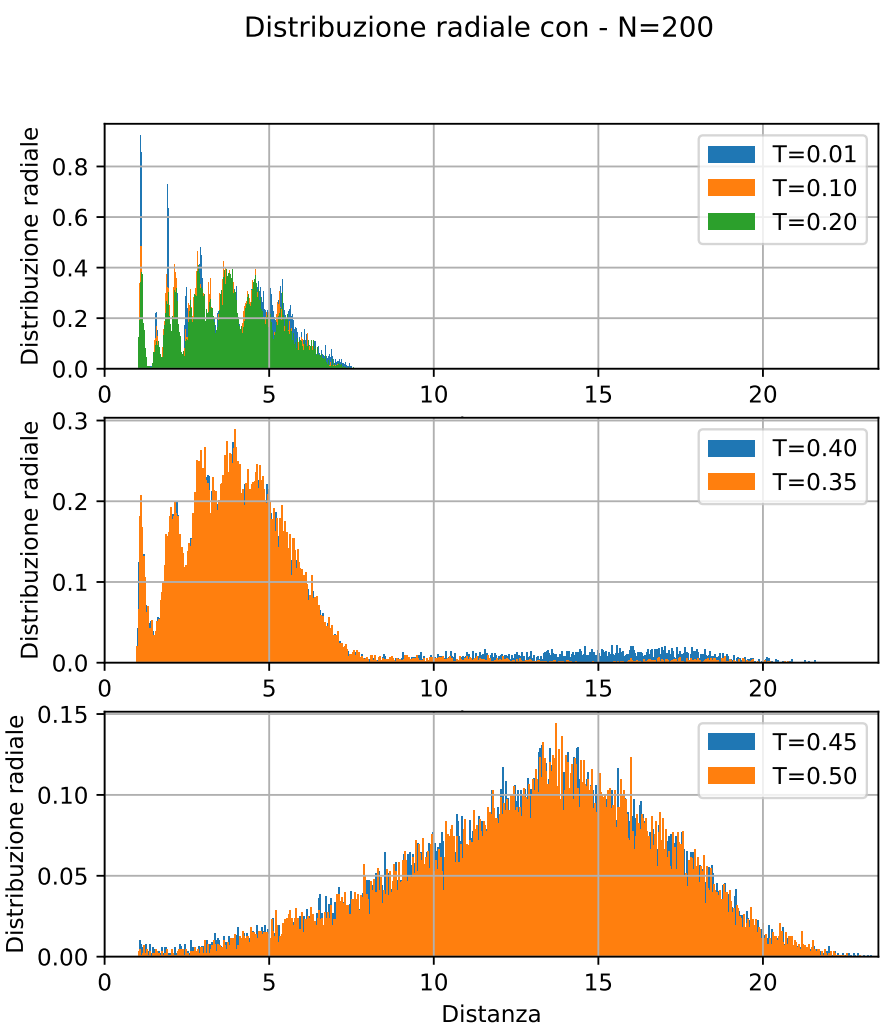


Figura 51: Grafici della distribuzione radiale con 200 atomi a varie temperature

ancora un tutt'uno. I picchi sono più bassi e larghi, questo suggerisce una maggiore mobilità degli atomi, quindi potrebbe essere una goccia di liquido.

A $T > T_V$ si ha un gas perfetto e tutte le particelle sono disperse nel volume.

4.12 Metodo Monte Carlo

Si è ora riprovato ad eseguire le simulazioni tramite un metodo Monte Carlo: in particolare si è utilizzata la catena di Markov applicando l'algoritmo di Metropolis-Hasting. Si è proceduto nel seguente modo:

1. Sono stati disposti gli atomi in un reticolo FCC.

2. Si è scelto di fare trials muovendo un solo atomo alla volta. Per semplicità si è scelto di scorrere in sequenza gli atomi, ovvero fare il primo trial sull'atomo con l'etichetta 1, il secondo trial sull'atomo con 2, etc.

3. Sono stati generati 3 numeri pseudocasuali tra $[0, 1]$ e sono stati moltiplicati per un valore indicato nel programma con `jump`. `jump` è lo spostamento massimo lungo una direzione che un atomo può avere, quindi i 3 numeri generati moltiplicati per `jump` sono lo spostamento dell'atomo scelto lungo i 3 assi. Eccetto nella fusione dove il valore viene definito arbitrariamente per aumentare i trial accettati, nel ricavare le isoterme si ha che `jump` è pari al passo reticolare del cristallo.

4. Si effettua un trial:

   - Se il trial viene accettato, salva la pressione nuova.
   - Se il trial viene rifiutato, si rimette l'atomo al suo posto e si salva di nuovo la pressione vecchia.

5. Dopo aver effettuato un trial su ciascun atomo della lista, si fa la media di tutte le pressioni: questa è una misura di pressione. Si ritorna poi al primo atomo e si ripete la procedura.

6. A seconda della temperatura si effettuano un numero differente di misure dato che si ha differente tasso di accettazione dei trials e un diverso tempo del sistema per "termalizzare":

   - Se la temperatura è $T > 1.0$ si effettuano 500 misure.
   - Se la temperatura è $0.7 < T \leq 1.0$ si effettuano 1000 misure.
   - Se la temperatura è $T \leq 0.7$ si effettuano 1500 misure.

   Si ricorda che il numero di trials è $N \times$ #misure.

7. Terminate le misure, queste venngono divise in due gruppi: la prima metà di misure fatte e la seconda metà.

   - Se le pressioni medie dei due gruppi sono uguali entro un certo errore, si salva il run.
   - Se le pressioni medie dei due gruppi sono diverse oltre questo errore, si cancellano le prime 50 misure, ne si fanno altre 50 e si applica nuovamente questo criterio.

   Questo criterio l'ho trovato necessario dato che non so quando il sistema termalizzi, per alcuni volumi e alcune temperature, il sistema necessita anche più di 2000 misure per termalizzare (a basse temperature ho visto run che arrivano a 30000 misure), quindi se non si applicasse questo metodo, tutte le misure sarebbero effettuate in un sistema che non ha termalizzato, di conseguenza tutte le pressioni ottenute e verrebbero contante con un peso troppo grande.

4.12.1 Isoterme

Sono stati effettuati studi su isoterme ottenute con il metodo Monte Carlo (MC). Si mostrano le isoterme ottenute con il metodo MC sovrapposte alle rispettive isoterme ottenute risolvendo le equazioni del moto (metodo MD).

Si nota che con 300 atomi (figura 52) i punti ottenuti con MC si sovrappongono perfettamente con le curve ottenute con MD, tuttavia si iniziano a vedere piccole differenze a $T = 0.8$. Effettuando

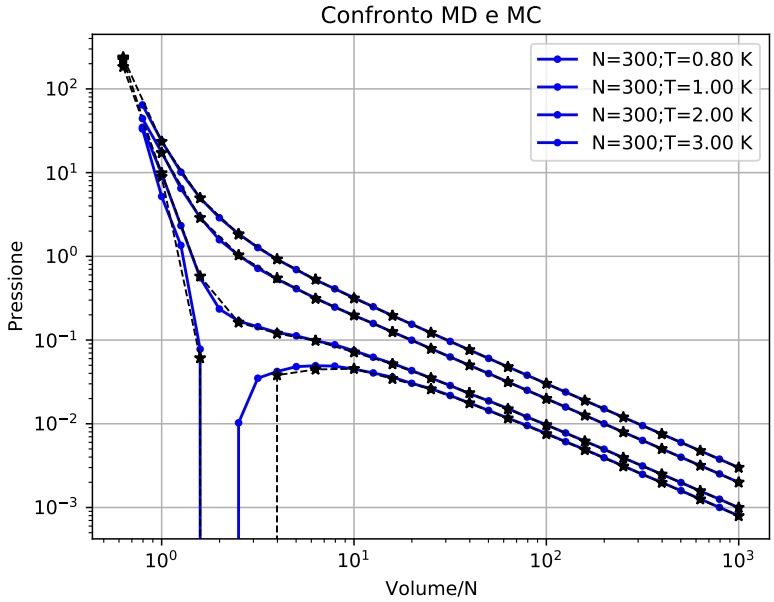


Figura 52: Isoterme ad alta temperatura con 300 atomi ottenute con MD (linee blu) con curve ottenute con MC (linee nere tratteggiate) sovrapposte.

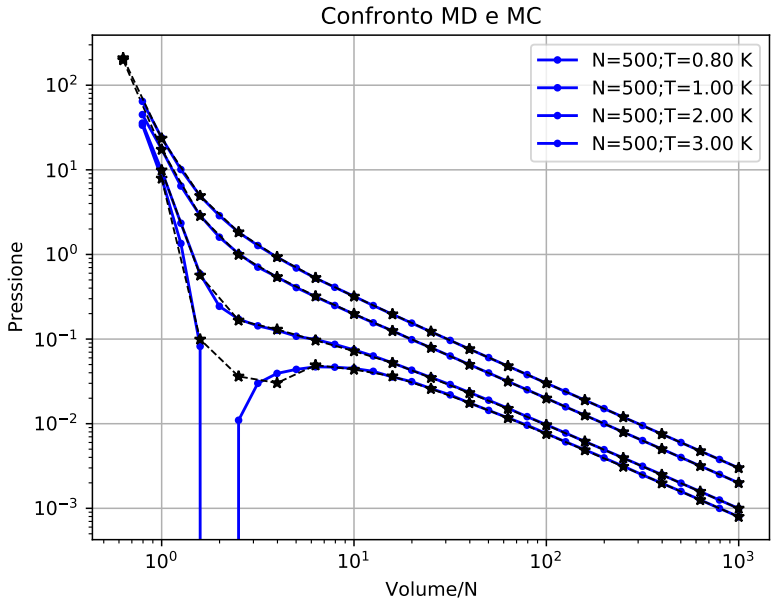


Figura 53: Isoterme ad alta temperatura con 500 atomi ottenute con MD (linee blu) con curve ottenute con MC (linee nere tratteggiate) sovrapposte.

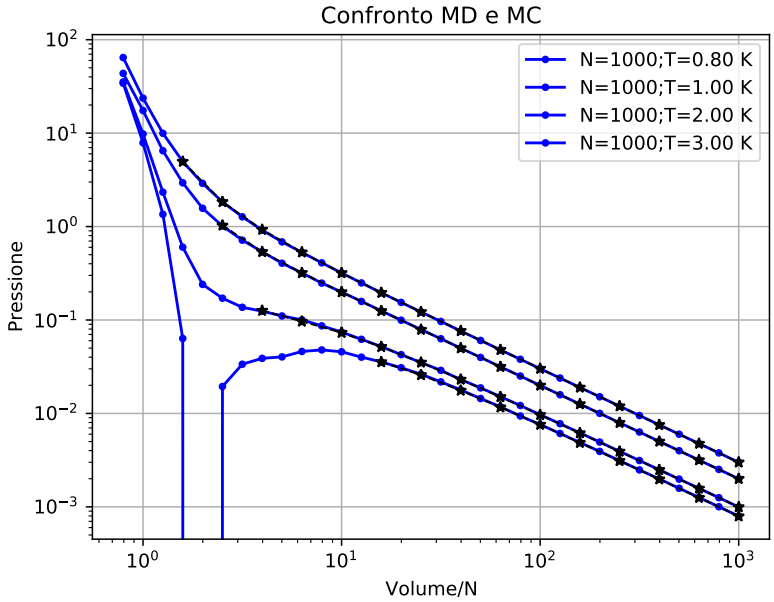


Figura 54: Isoterme ad alta temperatura con 1000 atomi ottenute con MD (linee blu) con curve ottenute con MC (linee nere tratteggiate) sovrapposte. Non ho purtroppo potuto finire i run dato che stava occupando molto tempo.

invece il confronto con 500 atomi (figura 53), si hanno le stesse considerazioni, ma a $T = 0.8$ si ha un andamento totalmente differente per $x <= 0.6$. Purtroppo però, per 1000 atomi (figura 54) la durata del run stava diventando eccessiva, quindi non ho finito di tracciare le isoterme; per i punti valutati ad alta temperatura, i risultati tra MD e MC sono consistenti.

4.12.2 Condensazione del gas

Per le stesse motivazioni per cui in precedenza ho fatto uno studio di condensazione, ora non ho fatto uno studio di fusione con il metodo MC, ma ho fatto uno studio di condensazione. Si mostrano ora i grafici energia-temperatura.

In questo caso non si nota una transizione di fase, si nota invece che le energie decrescono (al diminuire della temperatura) con un andamento fluttuante. Tuttavia a volumi piccoli si ha nuovamente accordo con la legge di Dulong Petit, come nel caso delle simulazioni MD.

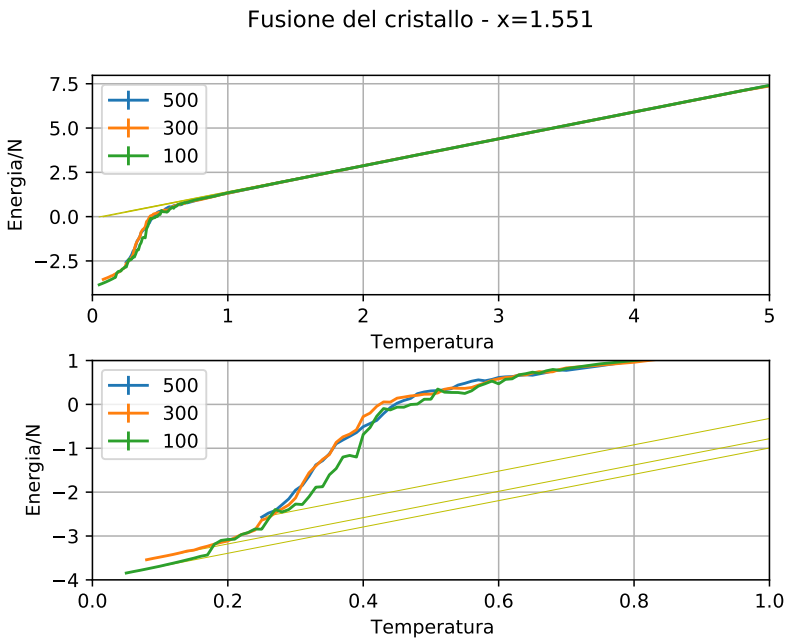


Figura 55: Simulazione di condensazione prima e solidificazione poi di un gas in un volume specifico $x = 1.551$

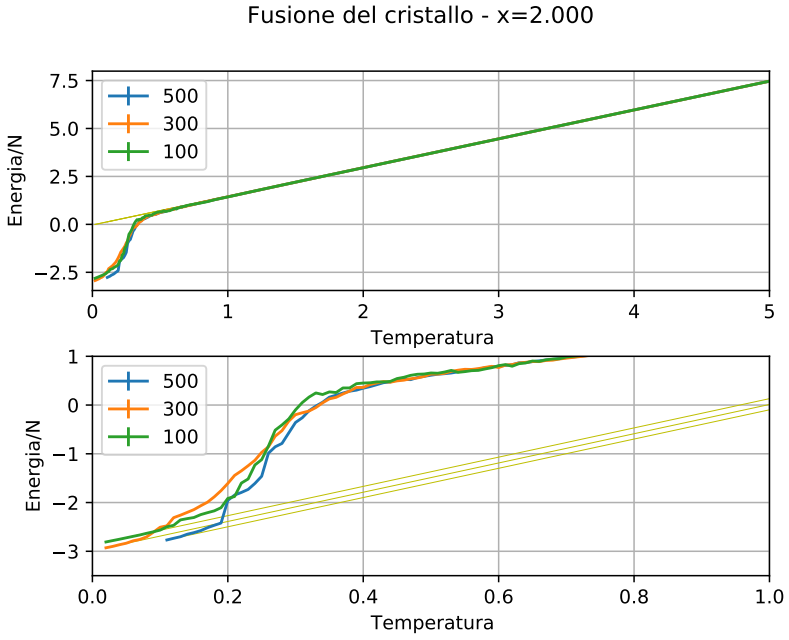


Figura 56: Simulazione di condensazione prima e solidificazione poi di un gas in un volume specifico $x = 2.0$

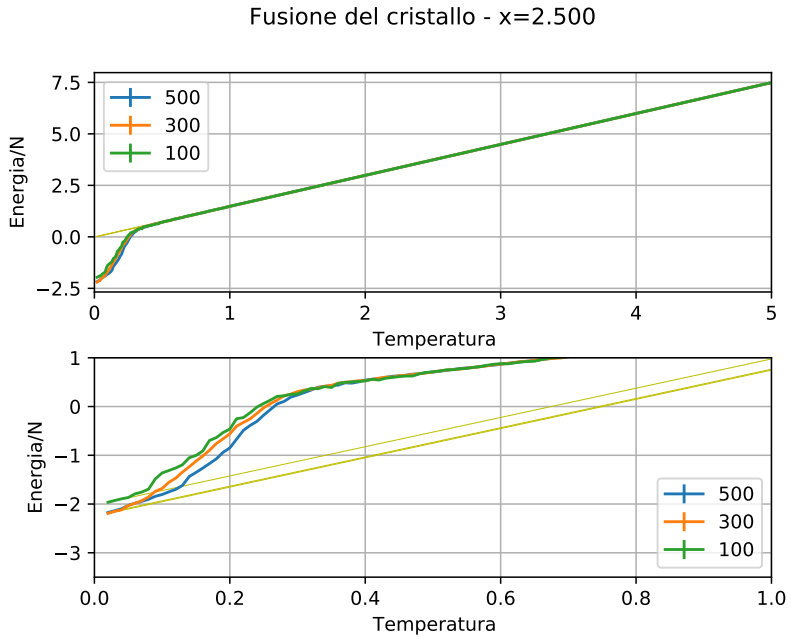


Figura 57: Simulazione di condensazione prima e solidificazione poi di un gas in un volume specifico $x = 2.5$

5 Diagrammi di Feynman

5.1 Calcolo delle tracce

Ho impostato il calcolo delle tracce per ottenere l'ampiezza quadra per lo scattering $e^- e^- \to e^- e^-$ tramite Maxima e ho confrontato il risultato con l'espressione fornita dal Prof. Zaffaroni durante le lezioni di Fisica Teorica I. I risultati sono coincidenti.

5.2 Calcolo delle ampiezza di elicità

Ho scritto un programma che calcolasse le ampiezze di diagrammi di Feynman, in particolare ho implementato tutte le funzioni indicate sulle note del corso.

Tutti i sottoprogrammi e le strutture che vengono utilizzati come libreria si trovano come modulo `strutturaDati` in `Ampiezze.f90`, le strutture dati sono:

- `momentum`: è una struttura con un array che va da 0 a 3, come dice il nome è il quadrimomento.
- `spinor`: è una struttura dati con un array di numeri complessi (per le 4 componenti dello spinore), un vettore di momenti (definito con `momentum`) e una variabile per indicare se lo spinore è barrato o no.
- `vector`: è una struttura dati per un bosone vettore; contiene un array di 4 numeri complessi per le 4 componenti del vettore e un vettore di momenti (definito anche questo con `momentum`).

Nella libreria sono presenti funzioni per inizializzare gli spinori associate a gambe esterne (`uSpinor`, `vSpinor`, `uBarSpinor` e `vBarSpinor`); quando gli spinori vengono inizializzati, queste funzioni chia-

mano le funzioni `xi_m` e `x_p` per generare gli autostati di elicità rispettivamente meno e più. I vertici vengono trattati con `SpinorPhoton_Spinor` (se le gambe esterne sono uno spinore e un fotone, la gamba interna è uno spinore) e `SpinorSpinor_Photon` (se le gambe esterne sono due spinori, la gamba interna è un fotone).

Il processo da considerare è scritto in un sottoprogramma in file a parte, ciascun file ha il nome del processo e contiene: cinematica, dinamica e ampiezza teorica (ottenuta su libri e appunti). Nel programma principale si chiama solo il processo da considerare e si impostano i momenti esterni.

Sono stati considerati 2 processi:

- $e^-e^+ \to \mu^-\mu^+$
- Scattering di Compton (per ora non sono ancora riuscito a farlo funzionare, probabilmente il bug si trova nel sottoprogramma `spinorspinor`).

5.2.1 Processo $e^-e^+ \to \mu^-\mu^+$

Si mostrano ora nelle figure 58 e 59 i grafici ottenuti dal conto numerico utilizzando le ampiezze di elicità e si confronta con il conto analitico ricavato sui libri.

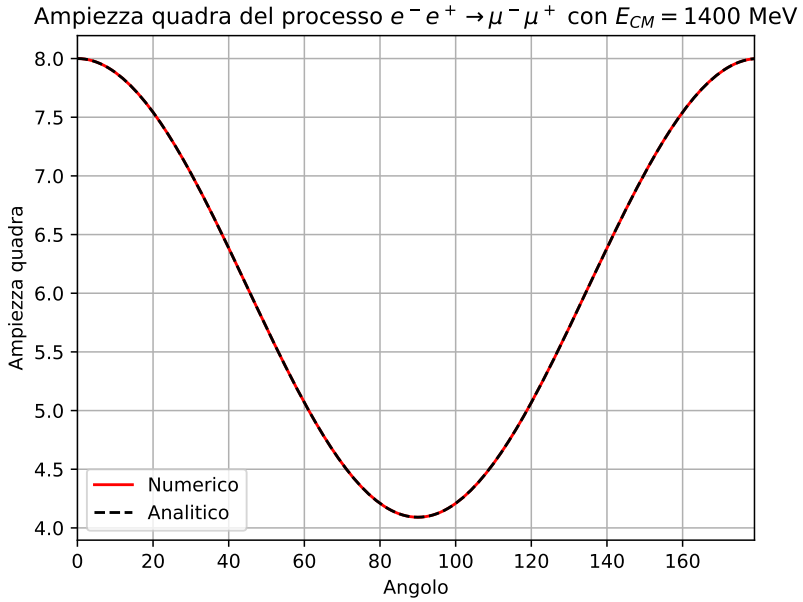


Figura 58: Ampiezza quadra con $E_{CM} = 1400$MeV, sull'asse delle ascisse è presente l'angolo rispetto alla direzione dell'elettrone nello stato iniziale. Per avere numeri non eccessivamente piccoli, si è posto anche e = carica dell'elettrone = 1

Detti p_1, p_2 i momenti entranti e k_1, k_2 i momenti uscenti, posto m_e = massa dell'elettrone, m_μ = massa del muone, $s = (p_1+p_2)^2 = (k_1+k_2)^2$, l'ampiezza modulo quadro al three level pesato sugli spin è:

$$\frac{1}{4}\sum_{spin}|M|^2 = \frac{8e^2}{s^2}\left[(p_1k_1)(p_2k_2) + (p_1k_2)(p_2k_1) + m_e^2(p_1p_2) + m_\mu^2(k_1k_2) + 2m_e^2m_\mu^2\right] \qquad (28)$$

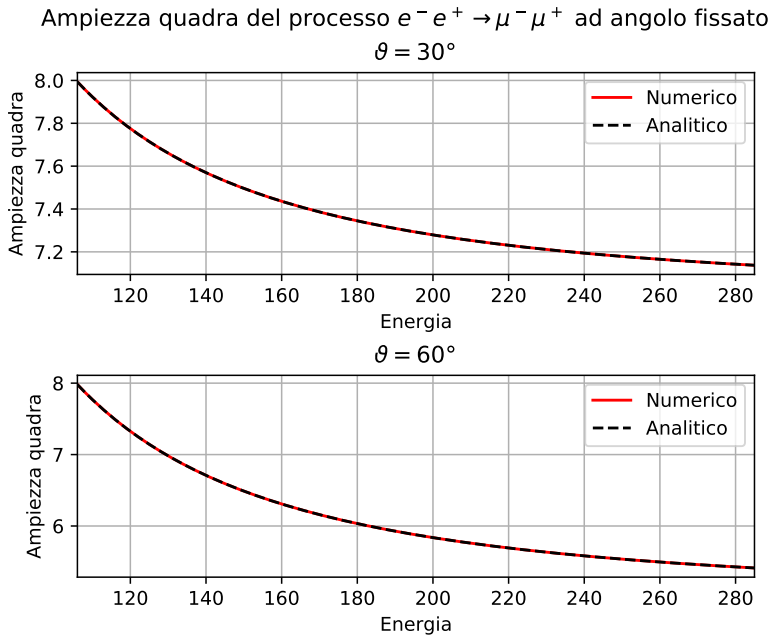


Figura 59: Ampiezza quadra ad angolo fissato variando l'energia (in MeV). I due angoli considerati sono $\vartheta = 30°$ e $\vartheta = 60°$. Per avere numeri non eccessivamente piccoli, si è posto anche $e =$ carica dell'elettrone $= 1$

5.2.2 Scattering di Compton

Detti p, p' i momenti (rispettivamente entrante ed uscente) dell'elettrone e k, k' i momenti del fotone (rispettivamente entrante ed uscente), posto $m =$ massa dell'elettrone, l'ampiezza modulo quadro al three level pesato sugli spin è:

$$\frac{1}{4}\sum_{spin}|M|^2 = 2e^4\left[\frac{pk'}{pk} + \frac{pk}{pk'} + 2m\left(\frac{1}{pk} - \frac{1}{pk'}\right) + m^4\left(\frac{1}{pk} - \frac{1}{pk'}\right)^2\right] \tag{29}$$

Non ho purtroppo un programma funzionante in questo caso, i risultati ottenuti con l'ampiezza di elicità non ritornano con l'espressione dell'equazione (29). Probabilmente c'è un errore nella contrazione di due spinori (propagatore).

6 Monte Carlo: prima parte del corso

Questi esercizi di applicazione del metodo di Monte Carlo sono scritti ed eseguiti in Python dato che sono presenti librerie molto comode per la generazione di grafici.

6.1 Integrazione di funzioni con il metodo Monte Carlo

Il primo esercizio consiste nella risoluzione del seguente integrale:

$$I = \int_0^1 \mathrm{d}x \int_0^1 \mathrm{d}y \, x^2 y^5 \tag{30}$$

Analiticamente, la soluzione è $I = \frac{1}{18}$.

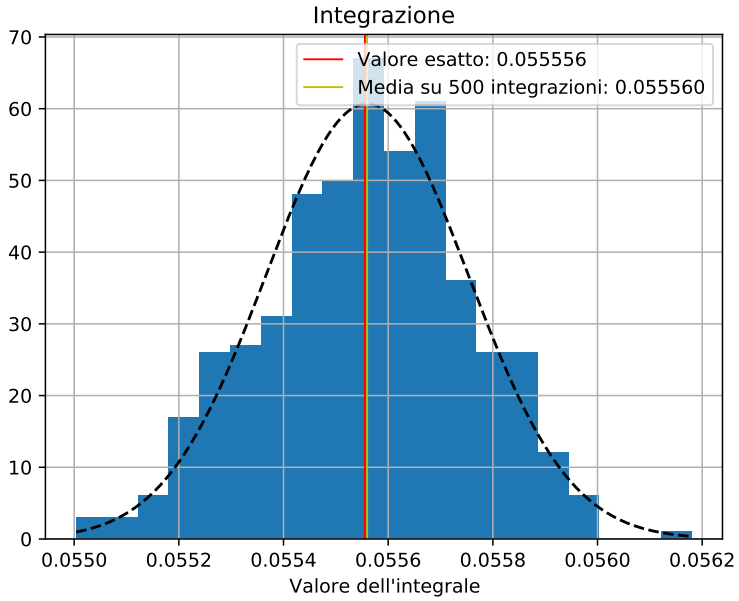


Figura 60: Svolgimento dell'integrale (30). Istogramma con 500 integrali svolti, ciascun integrale ha avuto $N = 400000$ punti.

Per valutare l'integrale 30 si è utilizzata l'equazione seguente:

$$S_N = \frac{1}{N} \sum_{i=1}^{N} f(x, y) \tag{31}$$

con $N = 400000$. Si è poi ripetuto 500 volte la stima di questo integrale e si sono inseriti i risultati in un istogramma. I risultati sono in figura 60; i risultati sono:

- Valore esatto dell'integrale: $I = \frac{1}{18} \approx 0.055555556$
- Valor medio di tutte le 100 stime dell'integrale: $\bar{f} \approx 0.055559803$
- Deviazione standard: $\sigma \approx 0.000193249$

Sulla figura è presente anche una gaussiana centrata in $\bar{f}$ e con deviazione standard σ.

6.2 Hit and miss

6.2.1 Esercizio 1

Il primo esercizio con il metodo *Hit and Miss* è la generazione di una distribuzione di probabilità che segue la funzione:

$$f(x) = \begin{cases} 12\left(x - \frac{1}{2}\right)^2 & \text{se } 0 < x < 1 \\ 0 & \text{altrimenti} \end{cases} \tag{32}$$

Questa è una distribuzione di probabilità dato che l'integrale su tutto lo spazio vale $I = 1$.

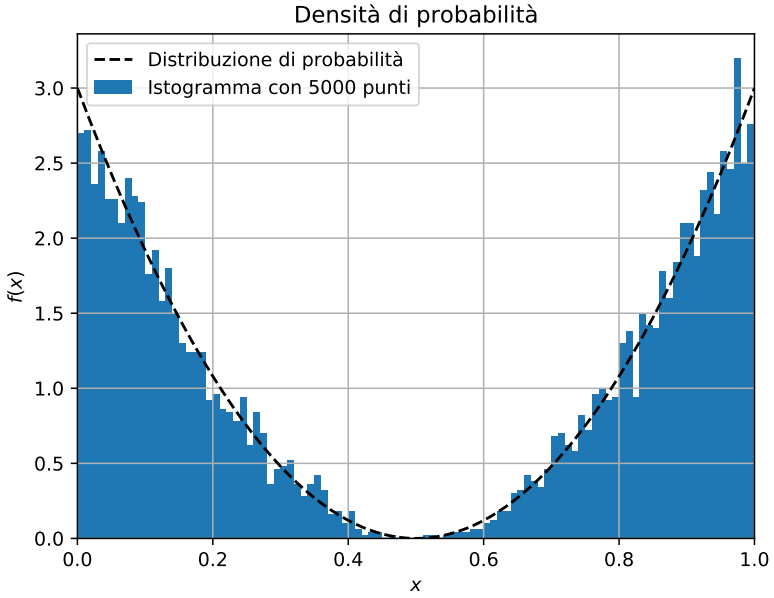


Figura 61: Distribuzione di probabilità che segue la funzione data.

Generando un numero $N = 5000$ punti, si ottiene la figura 61. Si nota che questa segue perfettamente la distribuzione di probabilità (32), ovvero la linea tratteggiata nera.

6.2.2 Esercizio 2

La risoluzione dell'esercizio 2 è identica alla risoluzione dell'esercizio 1, tuttavia l'unica differenza è che le x generate in modo casuale avranno un intervallo molto grande in modo da non perdere eccessivamente il contributo delle code.

6.3 Integrazione e generazione adattiva: caso unidimensionale

Ho implementato l'algoritmo descritto a lezione e ho provato ad integrare la distribuzione di probabilità utilizzato nell'esercizio precedente, ovvero ho provato ad integrare la funzione (32).

Si nota che il risultato dell'integrale calcolato con questo metodo adattivo può essere anche molto diverso dal valore reale, ma la media sulle 100 integrazioni risulta in un valore molto più vicino al

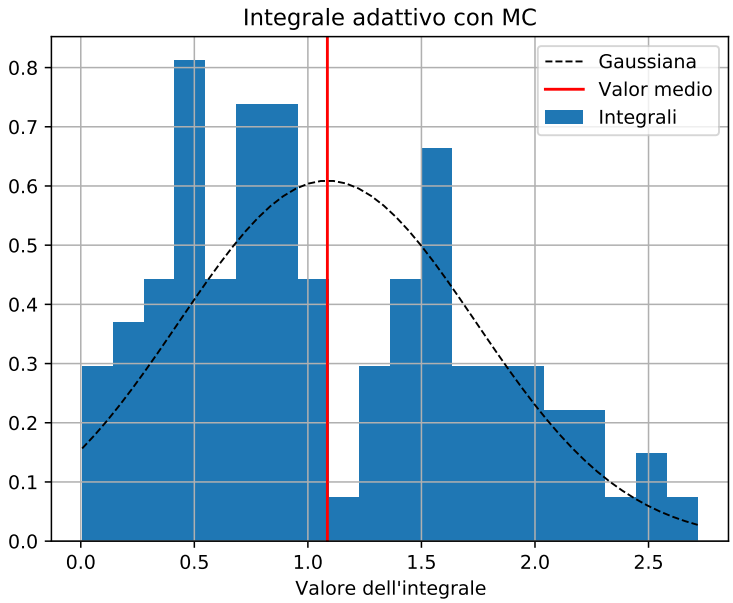


Figura 62: Integrazione dell'equazione (32). In tutto, per disegnare l'istogramma, sono state fatte 100 integrazioni.

valore analitico. Tuttavia i valori degli integrali mostrano una deviazione standard molto grande. Un modo per risolvere il problema è di mediare su più integrali.

7 Monte Carlo: seconda parte del corso

7.1 Decadimento radioattivo

Si ha un sistema di 1000 atomi con probabilità $\lambda = 0.01$ di decadere nello step temporale successivo.

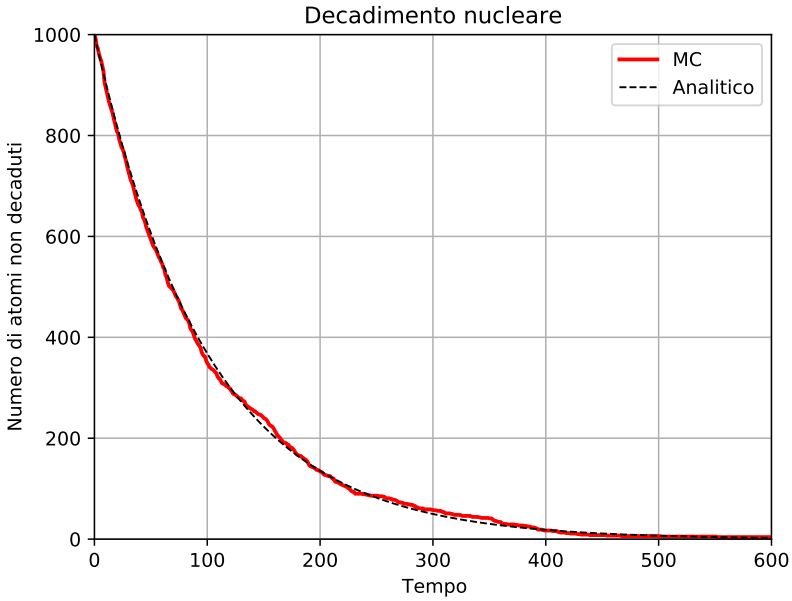


Figura 63: Decadimento di nucleare di 1000 atomi con probabilità $\lambda = 0.01$ di decadere per ciascuno step temporale. In rosso si ha la soluzione del sistema con il metodo MC, la linea tratteggiata nera è la soluzione dell'equazione differenziale che descrive il sistema.

Il programma parte da un numero iniziale di 1000 atomi non decaduti. Ad ogni step temporale viene chiesto a ciascun atomo se deve decadere: si genera un numero casuale p e lo si confronta con λ; se $p < \lambda$ l'atomo decade, altrimenti l'atomo non decade. È stato effettuato questo procedimento per ciascun atomo per 600 step temporali. La figura 63 mostra un confronto tra la risoluzione del sistema utilizzando il metodo Monte Carlo e la risoluzione dell'equazione differenziale associata.

L'equazione differenziale associata è:

$$\dot{N}(t) = -\frac{1}{\tau} N(t) \tag{33}$$

da cui si ottiene la soluzione:

$$N(t) = N_0 e^{-t/\tau} \tag{34}$$

dove:

- N_0 = numero iniziale di atomi
- $\tau = \frac{1}{\lambda}$

In figura 63 si nota che i due metodi di risoluzioni del sistema danno un risultato compatibile tra loro. Con il metodo MC si ha una piccola deviazione dal risultato analitico, ma è causato dal

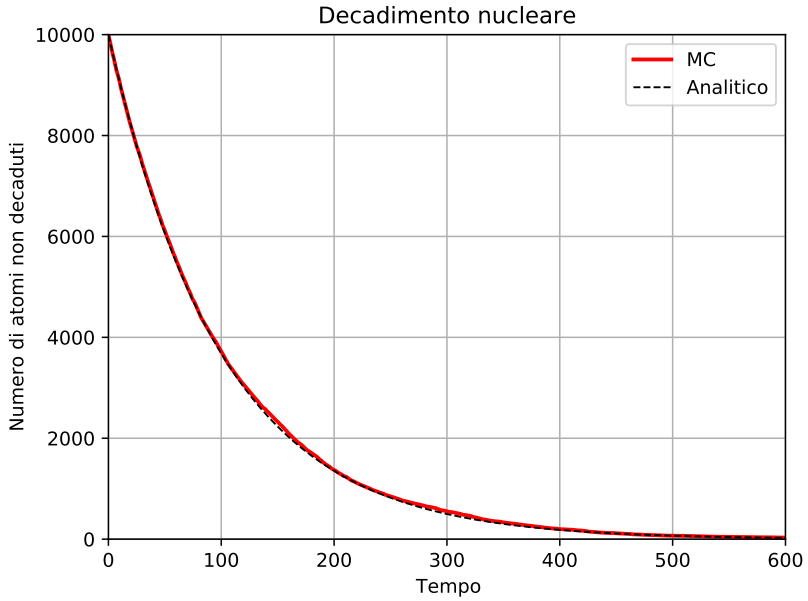


Figura 64: Decadimento di nucleare di 10000 atomi con probabilità $\lambda = 0.01$ di decadere per ciascuno step temporale. In rosso si ha la soluzione del sistema con il metodo MC, la linea tratteggiata nera è la soluzione dell'equazione differenziale che descrive il sistema.

fatto che si parte da un piccolo numero di atomi. Aumentando il numero di atomi si ottiene un andamento più liscio, come mostrato in figura 64

7.2 Previsioni del tempo

L'esercizio successivo è lo studio di un semplice sistema meteorologico; si hanno solo due stati: sole e pioggia. Le probabilità di passare da un tempo atmosferico al successivo sono riassunte nella tabella 2.

	Sole	Pioggia
Sole	0.9	0.5
Pioggia	0.1	0.5

Tabella 2: Tabella che rappresenta le probabilità di passaggio da tempo atmosferico al successivo. Se in un giorno c'è il sole, la probabilità che il giorno dopo ci sia sole è $P_{sole \to sole} = 0.9$, la probabilità che il giorno dopo ci sia pioggia è $P_{sole \to pioggia} = 0.1$

Questa tabella può essere scritta in una matrice di probabilità:

$$P = \begin{pmatrix} 0.9 & 0.5 \\ 0.1 & 0.5 \end{pmatrix} \tag{35}$$

Supponendo che nello stato iniziale ci sia sole, si può scrivere:

$$x_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \text{probabilità che ci sia sole} \\ \text{probabilità che ci sia pioggia} \end{pmatrix} \tag{36}$$

Allora si può definire lo step temporale successivo come:

$$x_1 = Px_0 \tag{37}$$

Ed in particolare

$$x_N = Px_{N-1} = P^N x_0 \tag{38}$$

Si ha un limite:

$$q = \lim_{N \to +\infty} x_N = \begin{pmatrix} 0.833 \\ 0.167 \end{pmatrix} \tag{39}$$

L'obiettivo dell'esercizio è di ritrovare questo limite. Il programma per effettuare la verifica è strutturato nel seguente modo:

- Si definiscono tutte le strutture dati, in particolare una matrice $2 \times N$ (`storico`) in cui si memorizza cumulativamente il numero di giornate di sole e di pioggia fino ad un certo giorno e una seconda matrice `frazione` $2 \times N$ che funge allo stesso scopo. `storico` viene utilizzato per tenere traccia degli stati di un singolo run. `frazione` viene utilizzato per tenere una media degli stati su tutti i run che vengono effettuati.
- In una simulazione, si parte impostando lo stato iniziale, ovvero si sceglie se durante il primo giorno si abbia sole o pioggia. Per il seguente giorno, si fa un trial per non cambiare tempo atmosferico.
- Dopo il trial, si incrementa in `storico` la casella corrispondente.
- Alla fine di un run, si somma `storico` alla matrice `frazione`
- Per ottenere il limite q bisogna fare la media su tanti run. Pertanto si rieffettua lo stesso run altre 99 volte reinizializzando `storico` all'inizio di ciascun run.
- Si dividono i valori in `frazione` per il numero di run. L'ultima coppia di valori è il limite q. Tanto maggiore è il numero di run, tanto migliore sarà l'approssimazione.

Descrivendo l'esempio concreto, si hanno i seguenti passaggi:

- Si inizializzano a zero le matrici `storico` e `frazione`.
- Si è scelto uno stato iniziale di pioggia (pioggia nel giorno 1).
- Si incrementa di 1 la casella pioggia nel giorno 1.
- Si sceglie con probabilità 50/50 se effettuare il trial con sole o pioggia (supponiamo che sia uscito pioggia).
- Si effettua un trial con $P_{pioggia \to pioggia}$. Ai fini dell'esempio, si suppone che il trial venga accettato.

- Il trial viene accettato, allora nel giorno 2 ci sarà pioggia. Pertanto, in `storico`, si prende il numero di giorni di pioggia (quindi 1), lo si incrementa di 1; invece per il giorno di sole, si copia nella casella del giorno 2 il contenuto della casella del giorno 1 (in questo caso 0). Se il trial viene rifiutato, nel giorno 2 ci sarà sole: si incrementerà così `storico` a favore di *sole*.
- Si effettua questo procedimento per 200 giorni.
- Alla fine dei 200 giorni, si somma `storico` a `frazione`.
- Si reinizializza a zero `storico` e si effettuano altri 99 run dello stesso tipo.
- Alla fine di tutti i run, si divide `frazione` per il numero di run. L'ultimo elemento in `frazione` è il limite cercato.

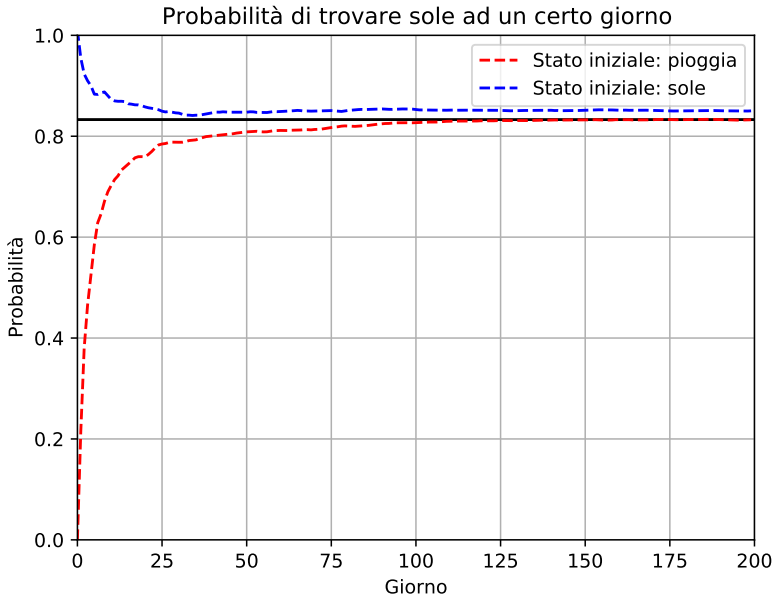


Figura 65: Probabilità di ottenere sole in un certo giorno dato uno stato iniziale. La probabilità si calcola come $P_{sole} = \frac{\#\text{giorni di sole}}{\#\text{giorni totali}}$. La linea nera orizzontale è $P = 0.833$

I risultati delle simulazioni sono mostrati in figura 65. Si ottiene esattamente il risultato che ci si aspetta, inoltre la frazione di giorni di sole non dipende dallo stato iniziale.

7.3 Diffuzione di un virus

Per lo studio della diffusione di un virus, si genera una griglia `lattice` di dimensione $N \times N$ in cui porre le persone; la persona nella casella (i, j) è infettata se `lattice[i][j] == 1` e può infettare altre adiacenti. Nella simulazione si è utilizzata una griglia con $N = 50$, la probabilità di infettare una persona adiacente è stata impostata a $P_{inf} = 0.3$. È stata scritta la funzione `infetta()` per determinare se infettare la persona adiacente selezionata viene o no infettata.

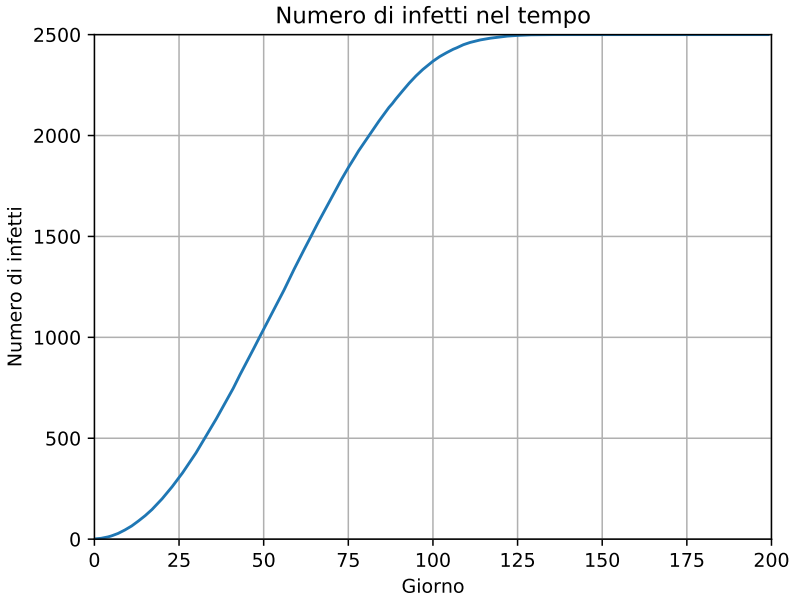


Figura 66: Numero di infetti nel tempo. Sono stati effettuate 20 simulazioni, il grafico è la media di questi 20 run.

Per ciascuna simulazione si parte dalla griglia `lattice` senza infetti; si infetta successivamente una persona casuale e si inizia il processo di diffusione dell'infezione. Per ciascun istante temporale si scorre la griglia alla ricerca di infetti; se si trova un infetto, si verifica che non sia stata appena infettata (le persone appena infettate vengono registrate nella matrice `appenaInfetto`), si prova ad infettare la persona adiacente (se non è già infetta). Dopo che è stata guardata l'ultima persona, si azzera `appenaInfetto`, pertanto le persone appena infettate sono ora in grado di infettare le persone adiacenti.

Sono state fatte 20 simulazioni e sono stati mediati i risultati, si mostra in figura 66 il risultato ottenuto.

Si nota un iniziale andamento di tipo esponenziale. Successivamente la curva ha iniziato ad acquisire una crescita lineare, finché tutte le persone presenti non sono state tutte infettate.

8 Modello di Ising

Per lo studio del modello di Ising, è stata creata una griglia $N \times N$ con $N = 40$. Si è inizializzato casualmente il reticolo con il 60% di spin up e 40% di spin down. Se non si fosse inizializzato in questo modo, il sistema non avrebbe scelto una delle due magnetizzazioni a bassa temperatura.

Successivamente si fa evolvere il sistema con 200 misure tramite il sottoprogramma `evolve`; questi step servono per far termalizzare il sistema prima di iniziare a fare le misure. Dopo la termalizzazione, si effettuano 2000 misure (sempre tramite `evolve`). Infine, dopo le 2000 misure, si salva il run.

Il sottoprogramma `evolve` implementa l'algoritmo di Metropolis-Hastings. In particolare: si sceglie uno spin casuale e lo si inverte, successivamente si fa un trial e si verifica se accettare o no la nuova configurazione, si salva la magnetizzazione della configurazione risultante (nuova se accettata o vecchia se rifiutata). Dopo $2 \times N \times N$ spin scelti a caso e dopo altrettante magnetizzazioni salvate, si effettua la media di tutte le magnetizzazioni salvate: questo costituisce una misura. L'intero processo viene effettuato 200 volte per il transitorio e 2000 volte per le misure effettive vere e proprie.

Ciascun punto visualizzato sul grafico è la media di tutte queste 2000 misure effettuate. L'errore (deviazione standard) è più piccolo del simbolo.

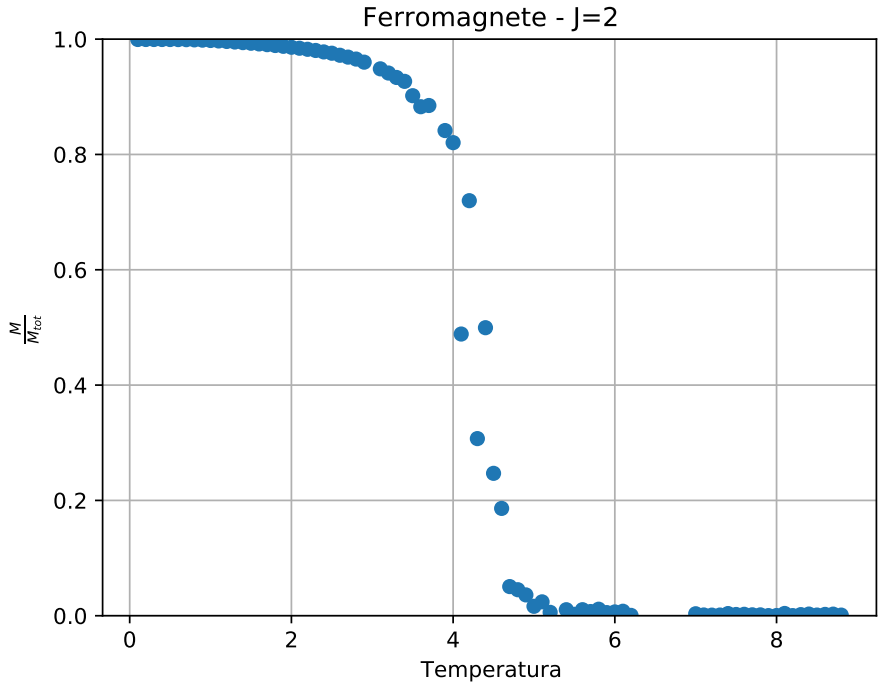


Figura 67: Magnetizzazione su magnetizzazione totale del modello di Ising bidimensionale

Avevo effettuato anche una simulazione con $j = 1$ e ho visto che la temperatura critica era minore, proprio come mi aspettavo.

A Gestione delle celle in dinamica molecolare

La scatola è suddivisa in cellette, ciascuna celletta contiene un certo numero di atomi. Questa struttura dati è stata modellata attraverso un oggetto $m \times m \times m$ dimensionale, ciascun elemento di questo oggetto contiene un puntatore ad una lista, la lista contiene gli atomi presenti nella celletta. Di seguito si mostra l'intestazione del modulo `liste`.

```
MODULE list_mod

    USE parametri

    ! Struttura che definisce un nodo
    TYPE listAtomsIndexType
        ! Indice dell'atomo
        INTEGER          :: atomIndex
        ! Puntatore che punta al nodo precedente
        TYPE(listAtomsIndexType), POINTER :: previous => NULL()
        ! Puntatore che punta al nodo successivo
        TYPE(listAtomsIndexType), POINTER :: next => NULL()
    END TYPE listAtomsIndexType

    ! Struttura che definisce un elemento dell'oggetto m*m*m
    TYPE listAtomsIndexTypePtr
        ! Puntatore alla testa della lista
        TYPE(listAtomsIndexType), POINTER :: head => NULL()
        ! Numero di elementi nella lista
        INTEGER     :: counter = 0
    END TYPE listAtomsIndexTypePtr

    ! Oggetto m*m*m dove ogni elemento e' associato ad una celletta
    TYPE(listAtomsIndexTypePtr) :: atomsInCellPtr(1:m,1:m,1:m)

    ! Matrice che contiene le coordinate delle celle di ogni atomo
    INTEGER :: atomsInCellCoordinates(1:3,1:nAtoms)

CONTAINS

   SUBROUTINE listAddHead(atomIndex,ix,iy,iz)

   SUBROUTINE listRmElement(atomIndex,ix,iy,iz)

   SUBROUTINE listCountAndCollect(interactAtomsIndexFilling,interactAtomsIndex,ix,iy,iz)

   SUBROUTINE fillInteractingArray(interactAtomsIndexFilling,interactAtomsIndex,ix,iy,iz)

   SUBROUTINE modCell(x,y,z)

END MODULE list_mod
```

Ciascuna celletta della scatola è associata ad una terna (x, y, z) che identifica la celletta stessa. Se chiamiamo (non presente nel codice) $smallBoxLength = boxLength/m$ = lunghezza di una celletta, le coordinate (i, j, k) corrisponderanno alla celletta con:

$$x \in [(i-1) \cdot smallBoxLength, i \cdot smallBoxLength]$$
$$y \in [(j-1) \cdot smallBoxLength, j \cdot smallBoxLength]$$
$$z \in [(k-1) \cdot smallBoxLength, k \cdot smallBoxLength]$$

Sono presenti due strutture dati derivate:

- `listAtomsIndexType` è il tipo che definisce il nodo di una cella. Esso contiene:
  - l'indice dell'atomo.
  - un puntatore che punta al nodo precedente (punta a `NULL()` se il nodo è il primo elemento della lista).
  - un puntatore che punta al nodo successivo (punta a `NULL()` se il nodo è l'ultimo elemento della lista).
- `listAtomsIndexTypePtr` è il tipo che punta alla testa di una lista e memorizza il numero di elementi in essa (punta a `NULL()` se la lista è vuota, ovvero se non ci sono atomi nella cella).

A far uso di queste due strutture dati è l'oggetto `atomsInCellPtr`, ovvero l'oggetto $m \times m \times m$ dimensionale in questione che indica quali atomi sono presenti in una data cella, in particolare è definito nel seguente modo: `TYPE(listAtomsIndexTypePtr) :: atomsInCellPtr(1:m,1:m,1:m)`. Infine `atomsInCellCoordinates` è una matrice $3 \times$ nAtomi che memorizza in quale cella si trova attualmente un atomo, la sua funzione è di evitare di dover scorrere ogni volta la `atomsInCellPtr` per verificare se un atomo si trova nella celletta. `atomsInCellCoordinates` viene aggiornata ogni volta che vengono effettuate operazioni sulla lista. Nelle prossime sezioni sono presenti le funzioni presenti nel modulo.

A.1 `listAddHead`

Aggiunge un nodo in testa: prende come ingresso l'indice dell'atomo `atomIndex` e le coordinate `(ix,iy,iz)` della celletta in cui l'atomo deve essere inserito. Dopo l'inserimento, si aggiorna il contatore del numero di nodi nella lista, ovvero si aggiorna `atomsInCellPtr(ix,iy,iz)%counter`. Successivamente si aggiorna anche `atomsInCellCoordinates`.

```
SUBROUTINE listAddHead(atomIndex,ix,iy,iz)

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: atomIndex, ix, iy, iz
    TYPE(listAtomsIndexType), POINTER :: oldHead => NULL()
    TYPE(listAtomsIndexType), POINTER :: newHead => NULL()

    newHead => listAtomsIndex(atomIndex)
    oldHead => atomsInCellPtr(ix,iy,iz)%head

    ! Se esiste una lista, riarrangia gli indirizzi
```

```
    IF (ASSOCIATED(oldHead)) THEN
        newHead%next => oldHead
        oldHead%previous => newHead
        atomsInCellPtr(ix,iy,iz)%head => newHead

    ! Altrimenti inizializza gli indirizzi e poi inserisci
    ELSE
        NULLIFY(newHead%next)
        NULLIFY(newHead%previous)
   END IF

   atomsInCellPtr(ix,iy,iz)%head => newHead

    ! Aggiorna il numero di elementi nella lista
    atomsInCellPtr(ix,iy,iz)%counter = atomsInCellPtr(ix,iy,iz)%counter + 1
    atomsInCellCoordinates(1,atomIndex) = ix
    atomsInCellCoordinates(2,atomIndex) = iy
    atomsInCellCoordinates(3,atomIndex) = iz

END SUBROUTINE listAddHead
```

A.2 `listRmElement`

Rimuove un nodo: prende come ingresso l'indice dell'atomo `atomIndex` e le coordinate `(ix,iy,iz)` della celletta in cui l'atomo si trova. La rimozione del nodo avviene collegando il nodo successivo al nodo precedente.

- Se il nodo da rimuovere è in testa, il puntatore al nodo precedente del nodo successivo punta a `NULL()`
- Se il nodo da rimuovere è in coda, il puntatore al nodo successivo del nodo precedente punta a `NULL()`

Dopo la rimozione, si aggiorna `atomsInCellPtr(ix,iy,iz)%counter`. Successivamente si aggiorna anche `atomsInCellCoordinates` ponendo nella cella `(0,0,0)` l'atomo, ovvero si rimuove l'atomo dalla scatola (N.B. le coordinate dell'atomo non vengono toccate, quindi solo la gestione delle liste legge come l'atomo fuori dalla scatola, il programma principale legge l'atomo come ancora all'interno della scatola).

```
SUBROUTINE listRmElement(atomIndex,ix,iy,iz)

    IMPLICIT NONE

    INTEGER, INTENT(IN) :: atomIndex,ix,iy,iz

    TYPE(listAtomsIndexType), POINTER :: previous => NULL()
    TYPE(listAtomsIndexType), POINTER :: actual => NULL()
    TYPE(listAtomsIndexType), POINTER :: next => NULL()

   actual => listAtomsIndex(atomIndex)
```

```
   previous => actual%previous
   next => actual%next

   IF (ASSOCIATED(previous)) THEN
      IF (ASSOCIATED(next)) THEN
         previous%next => actual%next
         next%previous => actual%previous
      ELSE
         NULLIFY(previous%next)
      END IF
   ! Se il nodo e' il primo elemento
   ELSE
      ! Se esiste una lista
      IF (ASSOCIATED(next)) THEN
         NULLIFY(next%previous)
         atomsInCellPtr(ix,iy,iz)%head => actual%next
      ! Se e' l'unico elemento
      ELSE
         NULLIFY(atomsInCellPtr(ix,iy,iz)%head)
      END IF
   END IF

   ! Aggiorna il contatore
   atomsInCellPtr(ix,iy,iz)%counter = atomsInCellPtr(ix,iy,iz)%counter - 1
   atomsInCellCoordinates(1,atomIndex) = 0
   atomsInCellCoordinates(2,atomIndex) = 0
   atomsInCellCoordinates(3,atomIndex) = 0

END SUBROUTINE listRmElement
```

A.3 listCountAndCollect

Il sottoprogramma conta il numero di atomi nella cella `(ix,iy,iz)` e riempie un array `interactAtomsIndex` che ne contiene gli indici a partire da `interactAtomsIndexFilling`. Il sottoprogramma successivamente incrementa `interactAtomsIndexFilling` del numero di atomi nella celletta `(ix,iy,iz)`, ovvero se per esempio `interactAtomsIndexFilling==5` prima dell'esecuzione del sottoprogramma e nella lista in `(ix,iy,iz)` ci sono 4 nodi, il sottoprogramma restituisce `interactAtomsIndexFilling=5+4=9` e `interactAtomsIndex` viene riempito dalle caselle dal 5 al 9.

```
SUBROUTINE listCountAndCollect(interactAtomsIndexFilling,interactAtomsIndex,ix,iy,iz)

   IMPLICIT NONE

    INTEGER, INTENT(INOUT) :: interactAtomsIndexFilling
    INTEGER, INTENT(OUT) :: interactAtomsIndex(1:nAtoms)
    INTEGER, INTENT(IN)  :: ix,iy,iz

    TYPE(listAtomsIndexType), POINTER :: ptr => NULL()

    ptr => atomsInCellPtr(ix,iy,iz)%head
```

```
    ! Esce dal DO WHILE quando ha finito di scorrere gli elementi della lista
    DO WHILE (ASSOCIATED(ptr))
        interactAtomsIndexFilling = interactAtomsIndexFilling + 1
        interactAtomsIndex(interactAtomsIndexFilling) = ptr%atomIndex
        ptr => ptr%next
    END DO

END SUBROUTINE listCountAndCollect
```

A.4 fillInteractingArray

Questo sottoprogramma decide quali caselle adiacenti alla casella `(ix,iy,iz)` valutare, in questo modo si dimezzano le operazioni necessarie per scorrere tutte le liste.

```
SUBROUTINE fillInteractingArray(interactAtomsIndexFilling,interactAtomsIndex,ix,iy,iz)

   USE parametri

   IMPLICIT NONE

   INTEGER, INTENT(INOUT) :: interactAtomsIndexFilling
   INTEGER, INTENT(OUT) :: interactAtomsIndex(1:nAtoms)
   INTEGER, INTENT(IN)   :: ix,iy,iz

   INTEGER            :: tx,ty,tz

   INTEGER :: i

   ! Colleziona atomi nelle celle vicine (Sono essere 13 blocchi di codice quasi
       identici)

   ! Blocco 1
   tx = ix
   ty = iy + 1
   tz = iz
   CALL modCell(tx,ty,tz)
   CALL listCountAndCollect(interactAtomsIndexFilling,interactAtomsIndex,tx,ty,tz)

   ! Altri 12 blocchi dello stesso codice con solo tx, ty e tz che cambiano

END SUBROUTINE fillInteractingArray
```

A.5 modCell

Si riportano gli indici `(x,y,z)` nel cubo di periodicità `m`.

```
SUBROUTINE modCell(x,y,z)

    USE parametri
```

```
   IMPLICIT NONE

   INTEGER, INTENT(INOUT) :: x,y,z

   IF (x>m) THEN
      x=x-m
   ELSE IF (x<1) THEN
      x=x+m
   END IF
   IF (y>m) THEN
      y=y-m
   ELSE IF (y<1) THEN
      y=y+m
   END IF
   IF (z>m) THEN
      z=z-m
   ELSE IF (z<1) THEN
      z=z+m
   END IF
END SUBROUTINE
```