Corso di Fisica Computazionale

- Prima Esercitazione -

QUANTIZZAZIONE SEMICLASSICA DEI LIVELLI VIBRAZIONALI

Cescato Matteo, Garbi Luca, Libardi Gabriele

Issue: 1

19 agosto 2020

Università degli Studi di Trento
Dipartimento di Fisica
Via Sommarive 14, 38123
Povo (TN), Italia

Indice

Abstract

In questa esercitazione viene studiato il moto vibrazionale di una molecola biatomica, il cui comportamento è modellizzato mediante il potenziale di Lennard-Jones. Ne vengono calcolati numericamente i livelli energetici in approssimazione semiclassica, ponendoli a confronto con i corrispondenti livelli ottenuti risolvendo analiticamente l'equazione di Schrödinger per un potenziale armonico che approssima quello esatto attorno al suo minimo. Dopo un breve richiamo teorico, viene descritto il codice utilizzato, in linguaggio C, ed infine vengono discussi i risultati ottenuti.

1 Quantizzazione semiclassica delle vibrazioni molecolari

Nella nostra descrizione di una molecola biatomica viene considerata l'approssimazione di Born-Oppenheimer: trascuriamo il moto degli elettroni attorno ai nuclei, in modo da poter separare le variabili del moto nucleare e le coordinate elettroniche nell'equazione di Schrödinger associata all'Hamiltoniana molecolare. In questo modo, possiamo considerare solamente il moto relativo dei due nuclei, governato da un potenziale $V(r)$, con r distanza tra essi. Per tenere conto sia del contributo attrattivo delle forze di van der Waals a grandi distanze, che di quello repulsivo dato dalle forze coulombiane e dal principio di esclusione, il potenziale considerato è quello di Lennard-Jones:

$$V(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]. \tag{1}$$

Il termine r^{-12} tiene conto del termine repulsivo, mentre r^{-6} descrive l'attrazione a lunga distanza. I parametri ε e σ rappresentano rispettivamente la profondità della buca di potenziale e il raggio della sfera che approssima l'atomo o la molecola in un modello a sfera rigida. Il potenziale, rappresentato in Fig.(1), presenta un minimo in $r_{min} = 2^{1/6}\sigma$, con una profondità pari a ε.

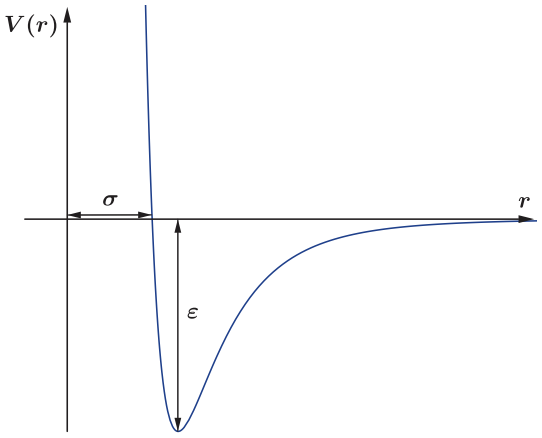


Figura 1: Potenziale di Lennard-Jones

Data la grande massa dei nuclei, possiamo disaccoppiare la componente vibrazionale del moto relativo da quella rotazionale, più lenta. Definita allora m la massa ridotta dei nuclei, i livelli di energia E_n si possono trovare come soluzione dell'equazione agli autovalori

$$\left[-\frac{\hbar^2}{2m}\frac{d^2}{dr^2} + V(r)\right]\psi_n(r) = E_n\psi_n(r)\,. \tag{2}$$

Nella nostra trattazione, il problema viene risolto sfruttando un'approssimazione semiclassica, ovvero si considera classicamente il moto degli atomi e vengono successivamente applicate delle regole di quantizzazione per determinarne l'energia. Le regole utilizzate sono quelle postulate nella vecchia teoria dei quanti da Bohr, Sommerfeld e Wilson, successivamente ritrovate nell'approssimazione WKB dell'equazione (2).

Classicamente, la distanza tra i nuclei atomici deve essere tale che l'energia totale rispetti la condizione

$$-\varepsilon \leq E < 0\,. \tag{3}$$

Chiamando p il momento relativo, l'energia complessiva può essere scritta come

$$E = \frac{p^2}{2m} + V(r) \quad \Longrightarrow \quad p(r) = \pm[2m(E - V(r))]^{1/2}$$

Ora, data una funzione d'onda $\psi(\mathbf{r},\, t)$, è sempre possibile scriverla come

$$\psi(\mathbf{r}, t) = \sqrt{n(\mathbf{r},\, t)}\exp\left(-\frac{S(\mathbf{r},\, t)}{i}\right),$$

dove $n(\mathbf{r},\, t) = |\psi(\mathbf{r},\, t)|^2$ è la densità di probabilità di trovare la particella in $\mathbf{r}$ al tempo t e, nel nostro caso, $S(\mathbf{r},\, t) \equiv S(E) = \oint k(r)\mathrm{d}r$ è l'azione (adimensionale) ad una data energia. Qui $k(r) \equiv p(r)/\hbar$ è il numero d'onda di de Broglie e l'integrale è calcolato su un ciclo completo di oscillazione (sia per $p(r)$ positivo che negativo). Dalle regole di quantizzazione sopracitate si ottiene che l'integrale d'azione calcolato sulle orbite quantizzate con energia permessa E_n deve essere pari a $2\pi(n + 1/2)$. Se indichiamo con r_{in} ed r_{out} i punti di inversione classici del sistema, ovvero quelli per cui $V(r) = E$, nel rispetto della condizione (3) e con $r_{in} < r_{out}$, allora, per semplice sostituzione, l'integrale d'azione diventa

$$S(E) = 2\left(\frac{2m}{\hbar^2}\right)^{\frac{1}{2}}\int_{r_{in}}^{r_{out}}\sqrt{E - V(r)}\,\mathrm{d}r\,,$$

dove il coefficiente 2 tiene conto del fatto che durante un'oscillazione la distanza tra i nuclei assume il valore r sempre 2 volte: una con p positivo ed una con p negativo. A questo punto, possiamo scrivere l'equazione per gli stati legati come

$$S(E_n) = 2\left(\frac{2m}{\hbar^2}\right)^{\frac{1}{2}}\int_{r_{in}}^{r_{out}}[E_n - V(r)]^{1/2}dr = 2\pi\left(n + \frac{1}{2}\right). \tag{4}$$

Per la risoluzione numerica risulta comodo esprimere l'azione in funzione delle grandezze adimensionali

$$\tilde{E} \equiv \frac{E}{\varepsilon}\,, \qquad x \equiv \frac{r}{\sigma}\,, \qquad \gamma \equiv \left(\frac{2m\sigma^2\varepsilon}{\hbar^2}\right)^{1/2}\,, \tag{5}$$

e del potenziale adimensionale

$$v(x) \equiv 4\left(\frac{1}{x^{12}} - \frac{1}{x^6}\right)\,, \tag{6}$$

in modo da riscrivere l'equazione (4) da risolvere come

$$\gamma \int_{x_{in}}^{x_{out}} \sqrt{\tilde{E}_n - v(x)}\,\mathrm{d}x - \pi\left(n + \frac{1}{2}\right) = 0\,. \tag{7}$$

2 Descrizione del codice utilizzato

Al fine di ricavare numericamente i valori di energia $\tilde{E}_n$ che soddisfano l'equazione (7) al variare del numero quantico n e per tre valori fissati di γ, si è realizzato un programma in linguaggio C. Nella funzione principale *main*, per ciascun valore di γ e ciascun valore di n ammesso, viene iterativamente richiamata la funzione *search*, che risolve l'equazione (7) mediante il metodo delle secanti. Tale equazione è del tipo $f(\tilde{E}) = 0$. Il calcolo della quantità $f(\tilde{E})$ è svolto dalla funzione *calcolaFunzione*, che a sua volta richiama la funzione *calcolaAzione* per valutare l'integrale

$$S(\tilde{E}) = \gamma \int_{x_{in}}^{x_{out}} \sqrt{\tilde{E} - v(x)}\,\mathrm{d}x\,. \tag{8}$$

Il calcolo dell'integrale richiede innanzitutto che vengano stabiliti gli estremi di integrazione x_{in} e x_{out}. A tale scopo, *calcolaAzione* richiama la funzione *trovaEstremi*, che risolve l'equazione in x

$$\tilde{E} - v(x) = 0\,. \tag{9}$$

Per ciascun valore fissato di $\tilde{E}$ tale che $-1 \leq \tilde{E} < 0$, le soluzioni di tale equazione sono 2, una a sinistra e una a destra del valore x_{min} per cui v raggiunge il minimo. L'algoritmo implementato in *trovaEstremi* ricava dapprima una stima grossolana di x_{in} e x_{out}, incrementando o decrementando iterativamente una variabile x, inizialmente posta pari a x_{min}, di una quantità dx sufficientemente piccola, finché non si ottiene

$$v(x) > \tilde{E}\,. \tag{10}$$

Considerando ad esempio la ricerca di x_{out}, se soddisfatta la condizione (10), la variabile x viene decrementata (una volta) della quantità dx e successivamente in-

crementata iterativamente di una quantità pari a $dx/2$, finché la condizione (10) non è nuovamente soddisfatta. A questo punto, l'incremento diventa pari a $dx/4$ e così via, fino a che l'incremento non diventa inferiore ad una soglia fissata. Analogamente avviene per la ricerca di x_{in}, con la differenza che in questo caso l'incremento è negativo. Con tale algoritmo viene ottenuta una prima stima delle soluzioni della (9). Una soluzione più precisa viene poi ottenuta con il metodo di Newton-Raphson. Si noti che la stima preliminare delle soluzioni è necessaria per garantire la convergenza dell'algoritmo di Newton-Raphson verso ciascuno dei due estremi d'integrazione. L'esistenza di due soluzioni e la forma della $v(x)$, infatti, possono dare luogo a situazioni patologiche. In particolare, non è garantito che l'algoritmo converga a x_{out} se viene eseguito il metodo di Newton-Raphson a partire da un punto $x > x_{min}$ qualsiasi. Nella funzione *trovaEstremi*, la valutazione del potenziale v e della sua derivata (esatta) in un punto x sono affidati alle funzioni *lennard_jones* e *der_lennard_jones* rispettivamente.
Una volta ottenuti gli estremi d'integrazione, in *calcolaAzione* viene valutata la funzione integranda in un numero fissato di punti equidistanti compresi nell'intervallo $[x_{in}, x_{out}]$. L'integrale viene infine stimato mediante la regola di Simpson cubica, implementata dalla funzione *integrale*.

Allo scopo di dare una stima dell'errore sui valori di energia trovati, nella funzione *search* viene utilizzato il metodo di bisezione per raffinare il risultato $\tilde{E}^*$ ottenuto mediante l'algoritmo delle secanti. Detti $\{\tilde{E}_k\}$ i valori esplorati dal metodo delle secanti, l'algoritmo di bisezione viene eseguito sull'intervallo compreso tra $\tilde{E}^*$ e l'ultimo valore $\tilde{E}_k$ per cui vale $f(\tilde{E}_k)f(\tilde{E}_{k+1}) < 0$. In questo modo è possibile assegnare al valore finale $\tilde{E}_n$ un'incertezza pari a metà dell'ampiezza dell'intervallo considerato nell'ultima iterazione del metodo di bisezione. In realtà, all'incertezza su $\tilde{E}_n$ contribuiscono anche gli errori commessi nella ricerca di x_{in} e x_{out}, e nel calcolo numerico dell'integrale (8). Con opportune scelte della condizione per terminare l'algoritmo di Newton-Raphson e del numero di punti considerati nell'integrazione, tali errori diventano però trascurabili rispetto a quello commesso nella funzione *search*, semplificando l'analisi delle incertezze.

Nel *main*, il numero di livelli energetici ammessi per ciascun valore di γ viene calcolato richiamando la funzione *numLivelli*, che restituisce la quantità

$$N = \left\lfloor \frac{\gamma}{\pi} \int_1^{+\infty} \sqrt{-v(x)}\, \mathrm{d}x - \frac{1}{2} \right\rfloor + 1 \,,$$

dove $\lfloor x \rfloor \equiv \max\{n \in \mathbb{N} \,|\, n \leq x\}$. L'integrale è stimato mediante le funzioni descritte in precedenza, fissando un valore di energia $\tilde{E}$ sufficientemente piccolo.

Al fine di ottimizzare il codice, vengono presi alcuni accorgimenti: nella funzione *search*, la ricerca del valore di energia associato al livello successivo inizia considerando come punto di partenza il valore relativo al livello precedente; nella funzione *trovaEstremi*, la ricerca dei nuovi x_{in} e x_{out} ha inizio fissando la variabile x uguale ai corrispondenti valori ottenuti per il livello precedente.

Parametri importanti per l'esecuzione dell'algoritmo, dichiarati come costanti nel codice, sono il numero di punti utilizzati nell'integrazione (NUM_PUNTI_INT) e i valori di soglia che stabiliscono l'accettabilità delle soluzioni delle due equazioni che vengono risolte. In particolare, l'algoritmo delle secanti per la ricerca dei livelli energetici si interrompe quando $|f(\tilde{E})| < tol\_E$; l'algoritmo di bisezione per la stima dell'errore sulle energie termina quando il doppio dell'errore è inferiore a TOL_E; la stima preliminare di x_{in} e x_{out} è accettata quando l'incremento diventa inferiore a tol_x; l'algoritmo di Newton-Raphson termina quando $|\tilde{E} - v(x)| < TOL\_X$.

I risultati ottenuti per ogni n e ogni γ sono salvati in un file di testo per la successiva elaborazione mediante l'ambiente MATLAB, con cui vengono tracciati alcuni grafici significativi. Il codice completo è riportato in Appendice A.2.

3 Discussione dei risultati e confronto con la teoria

Il problema è stato affrontato a livello numerico, in quanto l'equazione (7) non è risolvibile analiticamente, data la forma del potenziale di Lennard-Jones (1). Ciononostante, nulla ci vieta di considerare un'approssimazione di (1), che renda l'equazione di Schrödinger (2) analiticamente risolvibile e possa permettere di trovare delle corrispondenze con la soluzione numerica ottenuta con il potenziale esatto, almeno nel limite di piccole energie. Per fare ciò si sfrutta la soluzione nota dell'equazione (2) nel caso di un oscillatore armonico unidimensionale, il cui potenziale ha la forma $V(x) = \frac{1}{2}kx^2$ (con k costante). Lo spettro degli autovalori della sua Hamiltoniana è dato dalla relazione

$$E_n = \hbar\omega \left(n + \frac{1}{2}\right) \quad \text{con } n \in \mathbb{N},$$

dove $\omega \equiv \sqrt{k/m}$ (con m la massa).

Per ottenere un'analogia tra questo caso e il potenziale di Lennard-Jones, si approssima quest'ultimo ad una parabola attorno al suo minimo r_{min}, in modo da ottenere il termine quadratico tipico del potenziale armonico. Si considera dunque l'espansione in serie di Taylor al secondo ordine della funzione (1) attorno ad r_{min}:

$$V(r) = -\varepsilon + \frac{1}{2}\,\frac{72\,\varepsilon}{\sqrt[3]{2}\,\sigma^2}\left(r - \sqrt[6]{2}\sigma\right)^2 + o(r^3)\,. \tag{11}$$

Il potenziale approssimato si inserisce quindi nell'equazione (2), operando alcune opportune sostituzioni, in modo da ricondursi alla forma standard dell'Hamiltoniana dell'oscillatore armonico:

$$\left[-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + V(r)\right]\psi_n(r) = E_n\psi_n(r)\,,$$

$$\left[-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}r^2} - \varepsilon + \frac{1}{2}\frac{72\,\varepsilon}{\sqrt[3]{2}\,\sigma^2}\left(r - \sqrt[6]{2}\sigma\right)^2\right]\psi_n(r) = E_n\psi_n(r)\,,$$

$$\left[-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}r^2} + \frac{1}{2}\frac{72\,\varepsilon}{\sqrt[3]{2}\,\sigma^2}\left(r - \sqrt[6]{2}\sigma\right)^2\right]\psi_n(r) = (E_n + \varepsilon)\psi_n(r)\,,$$

$$\left[-\frac{\hbar^2}{2m}\frac{\mathrm{d}^2}{\mathrm{d}\tilde{r}^2} + \frac{1}{2}\,\tilde{k}\tilde{r}^2\right]\psi_n(r) = \mathcal{E}_n\psi_n(r)\,, \tag{12}$$

dove nell'ultimo passaggio sono state effettuate le seguenti sostituzioni:

$$\mathcal{E}_n \equiv E_n + \varepsilon\,, \quad \tilde{k} \equiv \frac{72\,\varepsilon}{\sqrt[3]{2}\,\sigma^2}\,, \quad \tilde{r} \equiv r - \sqrt[6]{2}\sigma \quad \Rightarrow \quad \frac{\mathrm{d}}{\mathrm{d}\tilde{r}} = \frac{\mathrm{d}}{\mathrm{d}r}\,. \tag{13}$$

Vista la forma dell'equazione (12), si può ricavare lo spettro degli autovalori sfruttando l'analogia con l'oscillatore armonico:

$$\mathcal{E}_n = \hbar\sqrt{\frac{\tilde{k}}{m}}\left(n + \frac{1}{2}\right).$$

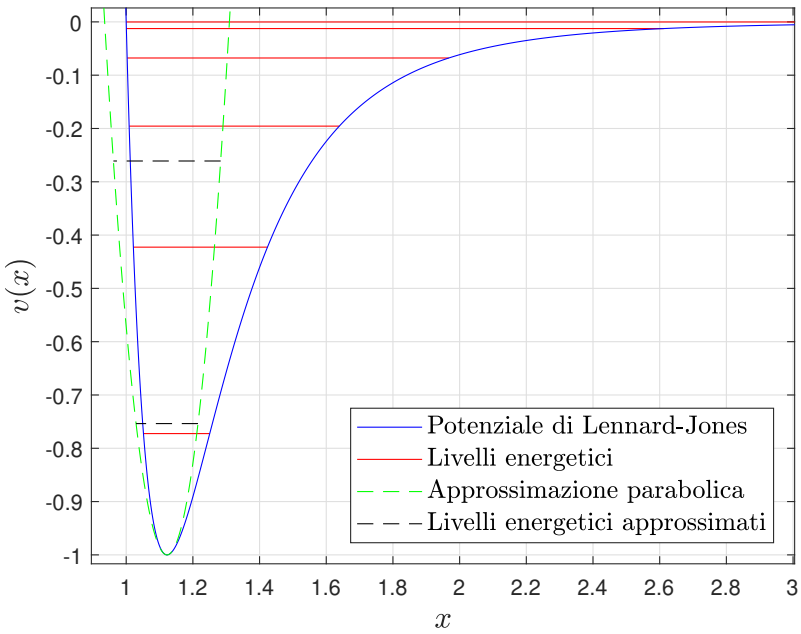
Infine, con alcuni passaggi algebrici, utilizzando le definizioni (5) e (13), si ottiene lo spettro degli autovalori in una forma (adimensionale) che può essere confrontata con lo spettro ricavato numericamente considerando il potenziale di Lennard-Jones e l'approssimazione semiclassica:

$$\tilde{E}_n = \left(\frac{12}{\sqrt[6]{2}\,\gamma}\left(n + \frac{1}{2}\right) - 1\right). \tag{14}$$

In Appendice A.1 sono tabulati i valori di energia $\tilde{E}_n$, calcolati numericamente, per le molecole H_2, HD e O_2, alle quali corrisponde un parametro γ pari a 21.7, 24.8 e 150 rispettivamente. Sono inoltre riportati i livelli energetici di energia negativa ottenuti considerando l'approssimazione armonica, calcolati mediante la formula (14). Di seguito, invece, sono presentati dei grafici che pongono a confronto i risultati ottenuti numericamente per il potenziale di Lennard-Jones esatto ed analiticamente considerando l'approssimazione armonica del medesimo. Da notare che, mentre in Appendice sono riportati tutti i livelli energetici di energia negativa per il potenziale approssimato, nei grafici sono rappresentati solamente i primi, ovvero quelli più vicini ai livelli dati dal potenziale esatto.

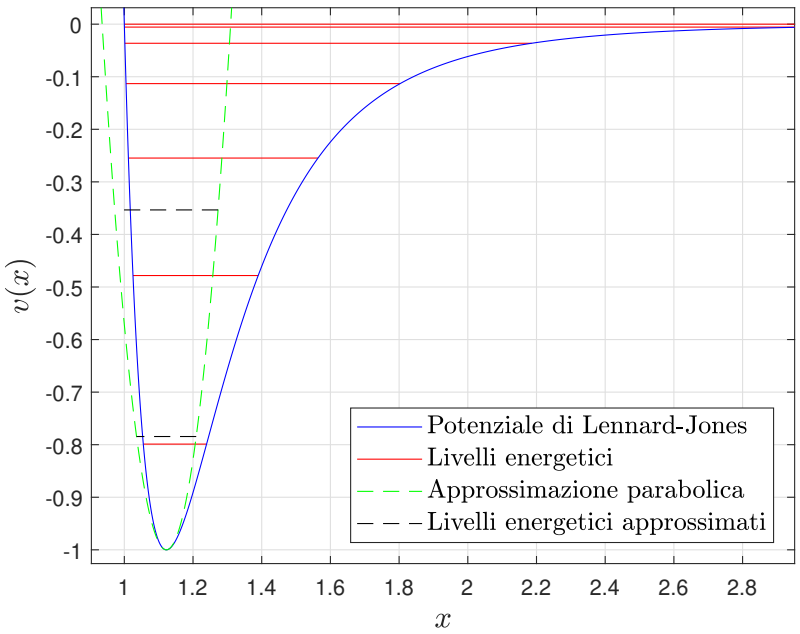
Dai grafici e dai valori tabulati si riscontra che per energie $\tilde{E} \lesssim -0.8$ il potenziale approssimato (11) rappresenta una buona approssimazione del potenziale (1), mentre per valori di energia maggiori l'approssimazione non risulta più valida, come ci si aspetta. Come ulteriore conferma della bontà dell'approssimazione semiclassica e dell'algoritmo progettato, si è infine modificato il codice sostituendo il potenziale di Lennard-Jones con il potenziale approssimato (11) (riscritto in forma adimensionale con le solite sostituzioni). I risultati numerici si sono rivelati essere un'ottima approssimazione di quelli esatti, ricavati mediante la relazione (14).

Figura 2: Risultati per $\gamma = 21.7$



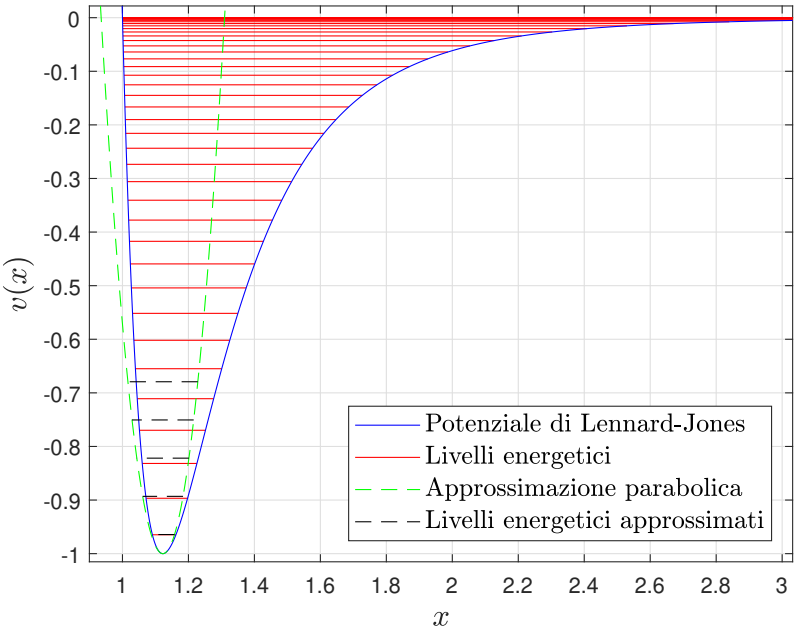
Sono rappresentati il potenziale di Lennard-Jones adimensionale (equazione (6)), i livelli energetici $\tilde{E}_n$ ricavati numericamente, l'approssimazione parabolica del potenziale e i primi livelli energetici ottenuti mediante tale approssimazione.

Figura 3: Risultati per $\gamma = 24.8$



Come in figura 2, ma qui i risultati sono relativi a $\gamma = 24.8$.

Figura 4: Risultati per $\gamma = 150$



Come in figura 2, ma qui i risultati sono relativi a $\gamma = 150$.

A Appendice

A.1 Livelli energetici per i tre valori di γ

	$\tilde{E}_n$ per il potenziale esatto			$\tilde{E}_n$ per il potenziale approssimato		
n	$\gamma = 21.4$	$\gamma = 24.8$	$\gamma = 150$	$\gamma = 21.4$	$\gamma = 24.8$	$\gamma = 150$
0	-0.7724485(4)	-0.7988766(4)	-0.9647637(4)	-0.753668	-0.784460	-0.964364
1	-0.4226381(3)	-0.4782143(3)	-0.8966711(4)	-0.261005	-0.353379	-0.893092
2	-0.1955785(5)	-0.2547127(3)	-0.8317031(4)			-0.821820
3	-0.0677432(4)	-0.1130917(3)	-0.7698011(4)			-0.750548
4	-0.0125431(4)	-0.0363412(4)	-0.7109063(4)			-0.679276
5	-0.00016888(3)	-0.0055596(3)	-0.6549584(4)			-0.608004
6		-0.00001063(2)	-0.6018965(3)			-0.536732
7			-0.5516583(3)			-0.465460
8			-0.5041809(3)			-0.394188
9			-0.45993999(3)			-0.322916
10			-0.4172500(3)			-0.251644
11			-0.3776649(3)			-0.180372
12			-0.3405768(3)			-0.109100
13			-0.3059170(2)			-0.037828
14			-0.2736155(3)			
15			-0.2436009(3)			
16			-0.2158007(3)			
17			-0.19014122(4)			
18			-0.16654685(5)			
19			-0.14494122(5)			
20			-0.12524637(6)			
21			-0.10738294(7)			
22			-0.09127013(7)			
23			-0.07682571(9)			
24			-0.0639660(1)			
25			-0.0526058(1)			
26			-0.0426586(1)			
27			-0.0340363(2)			
28			-0.0266494(2)			
29			-0.0204069(2)			
30			-0.0152166(3)			
31			-0.0109849(4)			
32			-0.0076164(3)			
33			-0.0050156(4)			
34			-0.003084230(7)			
35			-0.00172485(2)			
36			-0.00083768(9)			
37			-0.0003224(3)			
38			-0.000078984(1)			
39			-0.0000050942(8)			

A.2 Codice

Di seguito si riporta il codice in linguaggio C utilizzato.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// Costanti
#define PI 3.14159265358979323846 // Pi greco
#define E_MIN -1 // Valore di minimo del potenziale
#define X_MIN 1.1224620483093729814 // Punto di minimo del potenziale
#define NUM_PUNTI_INT 50000 // Numero di punti per calcolo integrali
#define tol_E 0.00001 // Tolleranza algoritmo delle secanti
#define TOL_E 0.000001 // Tol. algoritmo di bisezione per stima errore su E
#define TOL_X 0.00000000000001 // Tolleranza algoritmo di Newton-Raphson
#define dx_dx 0.2 // Incr. verso dx per stima preliminare zeri di E-v(x)
#define dx_sx 0.1 // Incr. verso sx per stima preliminare zeri di E-v(x)
#define tol_x 0.01 // Tol. per stima preliminare degli zeri di E-v(x)
const double GAMMA[3] = {21.7, 24.8, 150}; // Valori di gamma usati

// Prototipi funzioni
void search(short);
double calcolaAzione(double, short);
double calcolaFunzione(double, short);
void trovaEstremi(double);
int numLivelli(short);
double lennard_jones(double);
double der_lennard_jones(double);
double integrale(double*, double, int);

// Variabili globali
double E; // Energia
double dE; // Errore sull'energia
int N; // Numero livello
double x_estr[2]; // Estremi d'integrazione
double x_estr_prec[2]; // Estremi d'integrazione livello precedente


int main(){
    // File per salvataggio dati
    FILE *file;
    file = fopen("C:\\Users\\Admin\\Documents\\MATLAB\\esercizio1.txt", "w");

    short i;
    double numero_livelli;

    for(i = 0; i < 3; i++){
        printf("Gamma = %f\n\n", GAMMA[i]);
        fprintf(file, "%f %d %d\n", GAMMA[i], 0, 0);

        E = E_MIN + 2 * TOL_E;
        x_estr_prec[0] = X_MIN;
        x_estr_prec[1] = X_MIN;
```

```
        // Ciclo su tutti i livelli energetici
        numero_livelli = numLivelli(i);
        for(N = 0; N < numero_livelli; N++){
            search(i);
            printf("Livello␣%d:␣E␣=␣%1.11f␣+/-␣%1.11f;␣interv.␣integ.:␣"
                    "[%f,␣%f]\n", N, E, dE, x_estr[0], x_estr[1]);
            fprintf(file, "%f␣%f␣%f\n", E, x_estr[0], x_estr[1]);
        }

        printf("\n\n");
        fprintf(file, "\n");
    }

    fclose(file);

    return 0;
}


/* Cerca il livello energetico N-esimo per gamma = "GAMMA[index]"
mediante il metodo delle secanti e ne stima l'errore */
void search(short index){
    double funz[3], E_prec, temp, E1[3];
    E1[0] = -TOL_E;

    // Algoritmo delle secanti
    funz[1] = calcolaFunzione(E, index);
    E_prec = E - TOL_E;
    funz[0] = calcolaFunzione(E_prec, index);
    while(fabs(funz[1]) > tol_E){
        temp = E;
        E -= funz[1] * (E - E_prec) / (funz[1] - funz[0]);
        if (E >= 0)
            E = -TOL_E;
        E_prec = temp;
        funz[0] = funz[1];
        funz[1] = calcolaFunzione(E, index);

        /* Viene posto in "E1[0]" l'ultimo valore di energia trovato
        prima del cambio di segno di "calcolaFunzione(E, index)" */
        if(funz[1] * funz[0] < 0)
            E1[0] = E_prec;
    }

    // Stima dell'errore sull'energia - metodo di bisezione
    E1[2] = E;
    funz[2] = funz[1];
    funz[0] = calcolaFunzione(E1[0], index);
    while(fabs(E1[2] - E1[0]) > TOL_E){
        E1[1] = (E1[0] + E1[2]) / 2;
        funz[1] = calcolaFunzione(E1[1], index);
        if(funz[0] * funz[1] >= 0){
```

```
            E1[0] = E1[1];
            funz[0] = funz[1];
        }
        else{
            E1[2] = E1[1];
            funz[2] = funz[1];
        }
    }

    E = (E1[0] + E1[2]) / 2; // Livello energetico
    dE = fabs(E1[2] - E1[0]) / 2; // Stima dell'errore su "E"

    x_estr_prec[0] = x_estr[0];
    x_estr_prec[1] = x_estr[1];
}


double calcolaAzione(double E, short index){
    trovaEstremi(E);

    double h = (x_estr[1] - x_estr[0]) / (NUM_PUNTI_INT-1);
    double funz[NUM_PUNTI_INT];
    int i;

    // Calcolo integranda
    for(i = 0; i < NUM_PUNTI_INT; i++){
        funz[i] = sqrt(fabs(E - lennard_jones(x_estr[0] + i * h)));
    }

    return GAMMA[index] * integrale(&funz[0], h, NUM_PUNTI_INT);
}


double calcolaFunzione(double E, short index){
    return calcolaAzione(E, index) - (N + 0.5) * PI;
}


/* Trova gli estremi di integrazione,
   ossia i punti "x" tali che "lennard_jones(x) = E" */
void trovaEstremi(double E){
    double x, funz, dx;
    short i;

    for(i = 0; i < 2; i++){
        x = x_estr_prec[i];

        if(!i)
            dx = -dx_sx;
        else
            dx = dx_dx;

        // Mi avvicino allo zero entro "tol_x"
```

```
        do{
            if(lennard_jones(x) > E){
                x -= dx;
                dx /= 2;
            }
            x += dx;
        }while(fabs(dx) > tol_x);

        // Algoritmo di Newton-Raphson
        funz = E - lennard_jones(x);
        while(fabs(funz) > TOL_X){
            x  -= funz  / (-der_lennard_jones(x));
            funz = E - lennard_jones(x);
        }

        x_estr[i] = x;
    }
}


// Calcola il numero di livelli energetici ammessi per "GAMMA[index]"
int numLivelli(short index){
    return calcolaAzione(-TOL_E, index) / PI - 0.5 + 1;
}


// Calcola il potenziale di Lennard-Jones nel punto "x"
double lennard_jones(double x){
    return 4 * (1 / pow(x, 12) - 1 / pow(x, 6));
}


// Calcola la derivata del potenziale di Lennard-Jones nel punto "x"
double der_lennard_jones(double x){
    return 24 * (- 2 / pow(x, 13) + 1 / pow(x, 7));
}


/* Integrazione con regola di Simpson cubica
"*f" e' un puntatore al primo elemento di un vettore contenente
la funzione da integrare valutata in "numPunti" punti consecutivi
distanziati uno dall'altro di un intervallo "h".
Viene restituito l'integrale della funzione sull'intervallo coperto
dal vettore il cui primo elemento e' puntato da "*f",
stimato mediante la regola di Simpson cubica. */
double integrale(double *f, double h, int numPunti){
    double integ = (*f) + (*(f+numPunti-1));
    int i;

    for(i = 1; i < numPunti-1; i++){
        f++;
        if(i%2 == 0){
            integ += (*f) * 2;
```

```
        }
        else{
            integ += (*f) * 4;
        }
    }

    return integ * h / 3;
}
```