UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Implementazione di un algoritmo di minimizzazione soddisfacente il vincolo di unitarietà

Relatore

Prof. Paolo Umari

Laureando

Gian Lorenzo Marchioni

Anno Accademico 2022/2023

Introduzione

La teoria del funzionale della densità (DFT) è un metodo largamente usato per calcolare a principi primi le proprietà elettroniche e strutturali di sistemi atomici dalle molecole ai cristalli.
L'equazione di Schrödinger a molti corpi interagenti relativa ai gradi di libertà elettronici viene approssimata con le equazioni di Kohn-Sham (K-S) in cui gli elettroni sono trattati come particelle non interagenti. L'operatore Hamiltoniano di Kohn-Sham dipende infatti solamente dalla funzione densità di carica elettronica totale del sistema. Pertanto le equazioni di K-S sono di tipo autoconsistente e gli autostati dell'Hamiltoniana di K-S vengono identificati con gli orbitali elettronici.

L'energia del sistema descritto dalla DFT è invariante per trasformazioni unitarie del sottospazio degli orbitali occupati.
Questo rende interessante la ricerca di trasformazioni che abbiano particolare significato fisico. Per esempio, si possono cercare orbitali che siano più localizzati possibile o che soddisfino particolari simmetrie. Tale ricerca viene impostata come un problema di minimizzazione di un opportuno funzionale degli orbitali e la minimizzazione deve quindi soddisfare il vincolo di unitarietà.

Scopo del presente lavoro è presentare una soluzione numerica, implementando un algoritmo basato sui gradienti lungo le geodetiche nello spazio di Riemann [1]. L'algoritmo verrà utilizzato per il calcolo tramite differenze finite delle derivate degli orbitali. Si considererà un sistema dipendente da un parametro esterno, come la posizione di un atomo), e si calcolerà la derivata degli orbitali rispetto a tale parametro.
L'algoritmo sarà utilizzato per individuare la trasformazione unitaria che allinei il più possibile gli orbitali DFT calcolati per differenti valori di tale parametro. Va notato che tale problema è presente anche se gli orbitali considerati sono autostati dell'Hamiltoniana di K-S a causa delle degenerazioni in energia.

Cenni alla Density Functional Theory (DFT)

In questa sezione si riassume la teoria sottostante la soluzione iterativa utilizzata da *software* come *Quantum Espresso* (QE) per il calcolo delle funzioni d'onda molecolari.
Il riferimento principale è il testo "Heterogeneous Catalysts: Advanced Design, Characterization and Applications, II", in particolare il capitolo 23 [3].

La DFT si basa sulle seguenti assunzioni:

- Approssimazione di Born-Oppenheimer: la velocità dei nuclei è trascurabile rispetto a quella degli elettroni. È possibile trattare separatamente il moto di nuclei ed elettroni, e considerare solo gli ultimi come particelle quantistiche
- Primo teorema di Hohenberg e Kohn (H-K): nota la densità elettronica relativa allo stato fondamentale, $n_0(\vec{r})$, la funzione d'onda elettronica ψ_0 è unicamente definita. L'energia totale può essere espressa come funzionale della densità $E[n(\vec{r})]$
- Secondo teorema di H-K: la densità che minimizza $E[n(\vec{r})]$ corrisponde alla densità dello stato fondamentale $n_0(\vec{r})$
- Approccio di Kohn-Sham (K-S): il sistema "reale" di elettroni interagenti è approssimato con elettroni non interagenti. Attraverso la scelta di un opportuno potenziale "efficace" $v(\vec{r_i})$ tale sistema ha stessa densità $n_0(\vec{r})$ del caso reale

Si considera quindi l'equazione di K-S:

$$H_{KS}\phi_i(\vec{r}) = \left[-\frac{\hbar}{2m}\nabla_r^2 + v(\vec{r})\right]\phi_i(\vec{r}) = \epsilon_i\phi_i(\vec{r}) \tag{1}$$

Le cui soluzioni $\phi_i(\vec{r})$ sono le funzioni d'onda in regime di elettroni non interagenti, o funzioni d'onda di K-S. La densità elettronica corrisponde a:

$$n(\vec{r}) = \sum_{i=1}^{N} |\phi_i(\vec{r})|^2 \tag{2}$$

L'energia totale E è data da:

$$E[n(\vec{r})] = T[n(\vec{r})] + E_{\mathrm{H}}[n(\vec{r})] + E_{\mathrm{ext}}[n(\vec{r})] + E_{\mathrm{XC}}[n(\vec{r})]. \tag{3}$$

T è il termine cinetico degli elettroni non-interagenti; E_{H} è l'energia di interazione media fra elettroni, o energia di Hartree; E_{ext} è l'energia di interazione con il potenziale nucleare; E_{XC} è l'energia di scambio-correlazione. Quest'ultimo include tutti gli effetti non inclusi nel modello di K-S. Nei calcoli viene anch'esso approssimato con vari metodi la cui descrizione non è rilevante in questa tesi.
Gli autovalori di energia ϵ_i non sono direttamente coinvolti nel funzionale dell'energia del sistema.
Dalla 3 si ottiene un'ulteriore espressione per l'hamiltoniana H_{KS} utilizzando la derivata funzionale:

$$H_{KS} = \frac{\delta E[n(\vec{r})]}{\delta n(\vec{r})} = -\frac{\hbar}{2m}\nabla_r^2 + v_{ext}(\vec{r}) + \int d\vec{r}\,' \frac{n(\vec{r}\,')}{|\vec{r} - \vec{r}\,'|} + v_{XC}[n(\vec{r})] \tag{4}$$

H_{KS} dipende da $n(\vec{r})$ a causa del termine di scambio-correlazione v_{XC} e di potenziale di Hartree, ossia il termine integrale. Il termine di Hartree approssima l'interazione fra elettroni considerando un potenziale medio generato dalla loro distribuzione e integrando su tutto lo spazio.
Il termine v_{ext} rappresenta il potenziale "esterno" di interazione con i nuclei.

Allo stesso tempo, la densità dipende dalle soluzioni ϕ dell'equazione di K-S, il che rende necessario un approccio iterativo per la risoluzione.
Il metodo utilizzato da QE sarà discusso più avanti, ma solo qualitativamente.

Algoritmo di trasformazione unitaria

Utilizzando i fondamenti teorici della DFT e attraverso opportuni metodi numerici, è possibile risolvere le equazioni di K-S e ottenere la distribuzione spaziale degli elettroni. Entro la bontà delle approssimazioni alla base della DFT, gli autostati associati a questa distribuzione rappresentano gli orbitali degli elettroni, e sono caratterizzati da specifici autovalori di energia e da un numero di occupazione.

Denotando tali orbitali con $|\psi_i\rangle$, $i = 1, ..., N$, una generica trasformazione rappresentata da una matrice unitaria U, di dimensione $N \times N$, risulta agire su tali vettori di stato come

$$|\tilde{\psi}_i\rangle = \sum_{j=1}^{N} U_{ij}|\psi_j\rangle \tag{5}$$

Si considerano, in questo caso, funzioni d'onda orbitali reali, per un motivo legato alla loro espansione in serie che sarà discusso nella prossima sezione.
Questo fa sì che la matrice $\hat{U}$ sia quindi di reali, e che la relazione di unitarietà corrisponda a:

$$U^{\mathsf{T}}U = I \tag{6}$$

Vista l'arbitrarietà di U non è possibile usare direttamente il metodo della differenza finita per calcolare le derivate degli orbitali.
Infatti, anche scegliendo come orbitali gli autostati dell'operatore Hamiltoniano (di Kohn-Sham) rimane l'ambiguità dovuta a possibili stati degeneri in energia.
Indicando con $|\tilde{\psi}_i\rangle$ gli orbitali del sistema perturbato, l'obiettivo è individuare la matrice di trasformazione unitaria (reale) $\hat{U}$ tale da minimizzare il seguente funzionale degli orbitali, detto *cost function*:

$$\mathcal{J}(U) := \sum_{i,j}^{N} (-1)\langle\psi_i|\hat{U}_{ij}|\tilde{\psi}_j\rangle \tag{7}$$

Notiamo che per trovare $\hat{U}$ non occorre conoscere gli orbitali ma basta la matrice di *overlap* fra orbitali perturbati e non perturbati. $\hat{O}$:

$$O_{ij} := \langle \psi_i | \tilde{\psi}_j \rangle \tag{8}$$

Infatti è possibile scrivere, in quanto l'elemento $\hat{U}_{ij}$ è uno scalare:

$$\langle \psi_i | \hat{U}_{ij} | \tilde{\psi}_j \rangle = \hat{U}_{ij} \langle \psi_i | \tilde{\psi}_j \rangle = \hat{U}_{ij} O_{ij} \tag{9}$$

Di conseguenza la relazione 7 può essere scritta come

$$\mathcal{J}(U) = -\sum_{i,j}^{N} O_{i,j} \hat{U}_{i,j} \tag{10}$$

Per effettuare la minimizzazione, con il vincolo di unitarietà per $\hat{U}$, si fa riferimento al metodo *"basic riemannian steepest descent algorithm"* descritto in articolo [1], *Table 1.*
La matrice U viene inizializzata all'identità, $U_0 = I$, mentre il gradiente nello spazio euclideo della $\mathcal{J}$ si nota essere la matrice di *overlap* cambiata di segno:

$$\Gamma_k := \frac{\partial \mathcal{J}(U_k)}{\partial U_k} = -O_k \tag{11}$$

In cui l'intero k indica lo *step* di iterazione. Segue che il gradiente nello spazio di Riemann ha forma:

$$G_k := U_k O_k^{\mathsf{T}} - O_k U_k^{\mathsf{T}} \tag{12}$$

Si calcola quindi la matrice di rotazione P_k, corrispondente allo spostamento lungo la geodetica, sulla varietà di Riemann, opposta alla direzione del gradiente

$$P_k := exp(-\mu G_k) \approx \sum_{m=0}^{T} \frac{(-\mu)^m}{m!} (G_k)^m \tag{13}$$

In cui μ è un parametro reale positivo che controlla la velocità di convergenza. Nell'implementazione l'esponenziale viene approssimato in serie di Taylor fino all'ordine $T = 30$. Si aggiorna poi la U_k moltiplicandola per la rotazione P_k:

$$U_{k+1} = P_k U_k \tag{14}$$

La matrice di rotazione degli orbitali $\hat{U}$ stimata è ottenibile analiticamente come inversa della trasposta della matrice di *overlap* iniziale O. Al termine dell'algoritmo, si effettua una verifica di tale relazione analitica:

$$\hat{U} = \left(O^{\mathsf{T}}\right)^{-1} \implies \hat{U} O^{\mathsf{T}} = I \tag{15}$$

Effettuare questa operazione, tuttavia, può richiedere maggiore costo computazionale rispetto alla minimizzazione *steepest descent* per matrici di dimensione elevata.

Nello pseudo-codice che segue viene illustrata in sintesi l'implementazione dell'algoritmo.
Con la notazione nel titolo si intende che la matrice O e il suo ordine N vengano passati come argomenti all'algoritmo.
Nel codice C utilizzato tutte le matrici sono, inoltre, salvate in array monodimensionali di lunghezza $N \times N$.
In tutte le formule che seguiranno si sottintende l'utilizzo della funzione `cblas_dgemm` ogni volta che è presente un prodotto fra matrici, indicato con "$\times$". È possibile chiamare tale funzione includendo nel codice "`#include<mkl.h>`" e utilizzando un compilatore Intel.

Algorithm 1 Implementazione Basic Riemannian SD - algo(O, N)

$T = 30$ ▷ Ordine di espansione di Taylor
$\mu = 0.1$ ▷ Parametro di convergenza
Alloca ed inizializza le matrici U, G, P, I a matrici zero di dimensioni $N \times N$
Imposta gli elementi diagonali di U e I a 1
Alloca ed inizializza le matrici ausiliarie A, B, C, K, L a matrici zero di dimensioni $N \times N$
for step = 1 fino a n_{steps} **do**
    Calcola $A = O \times U^\top$ e $B = U \times O^\top$
    Calcola $G = B - A$ ▷ Calcolo del gradiente
    Inizializza $P = I$ e $K = I$
    $f = 1$
    for $m = 1$ fino a T **do** ▷ Espansione in serie per P
        Calcola $C = K \times G$
        $f = f \times m$
        Aggiorna $P = P + \frac{(-\mu)^m}{f} \times C$
        $K = C$
    end for
    Calcola $L = P \times U$
    Aggiorna $U = L$
end for
Verifica $U \times O^\top = I$ e copia U in O
Libera la memoria allocata per tutte le matrici

Applicazione: Orbitali di valenza molecolari con Quantum Espresso

Il programma utilizzato per i calcoli di DFT è *Quantum Espresso* (QE), in particolare il pacchetto `PWscf`. Si basa sull'espansione delle funzioni d'onda in serie di onde piane, e sull'utilizzo di pseudo-potenziali per ciascuna specie atomica.
Questi ultimi sostituiscono il potenziale reale del nucleo e degli elettroni interni, semplificando i calcoli: in questa approssimazione gli elettroni interni sono soggetti ad un potenziale costante, mentre solo per gli elettroni di valenza si utilizza l'Hamiltoniana di K-S illustrata precedentemente.
L'espansione in onde piane viene troncata ad una certa soglia energetica, ed il ciclo iterativo viene effettuato valutando le funzioni d'onda per determinati punti $\vec{k}$ del reticolo reciproco, secondo una griglia all'interno della zona di Brillouin specificata dall'utente.

Per i casi che seguono, ossia molecole singole, è sufficiente $\vec{k} = 0$. Questo semplifica notevolmente l'espressione per le funzioni d'onda, scritta in 16, e garantisce che siano reali.
Considerando una funzione d'onda $\phi_i(\vec{r}, \vec{k})$, con momento cristallino $\vec{k}$, la sua espansione in onde piane può essere scritta come:

$$\phi_i(\vec{r}, \vec{k}) = \sum_{\vec{G}} C_i(\vec{G}) e^{i\vec{G}\cdot\vec{r}} \tag{16}$$

In cui $\vec{G}$ sono vettori del reticolo reciproco; $C_i(\vec{G})$ sono i coefficienti di Fourier corrispondenti a ciascun $\vec{G}$ per l'i-esimo orbitale con momento $\vec{k}$, che non compare essendo identicamente nullo.

Centralmente, `PWscf` opera attraverso un approccio iterativo noto come ciclo SCF (Self-Consistent Field), che inizia con una stima della densità elettronica da cui si deduce il potenziale di Hartree. L'equazione di Kohn-Sham viene poi risolta per ottenere le funzioni d'onda elettroniche aggiornate e, di conseguenza, la densità elettronica aggiornata. Questo procedimento è ripetuto fino a quando l'energia del sistema converge entro un determinato limite: durante le procedura, l'energia totale viene calcolata ad ogni passaggio e confrontata l'iterazione precedente. Se la differenza tra questi due valori consecutivi è minore di un determinato valore limite (`conv_thr`), il sistema ha raggiunto una soluzione auto-coerente in termini di energia.

Lo si esegue fornendogli un file di *input*, estensione `.in`, in cui sono presenti:

- Comandi che specificano il tipo di calcolo da effettuare
- Le *directory* in cui salvare il *file* di *ouput*, e in cui sono presenti i *file* con gli pseudo-potenziali, con estensione `.upf`
- Parametri relativi al reticolo, come il tipo e la dimensione della cella di Bravais
- Il numero di atomi e le specie atomiche
- L'energia di *cut-off* per l'espansione in serie di onde piane
- Il numero di bande di valenza, `nbnd`
- Un parametro `mixing_beta`, utilizzato per aggiornare la densità includendo una combinazione lineare con la densità precedente
- Il parametro di covergenza `conv_thr`
- La configurazione della molecola, ossia le coordinate cartesiane di tutti gli atomi
- Il metodo di *sampling* dei punti del reticolo reciproco: nei casi che seguiranno si utilizzerà *gamma*, solo nell'origine, sufficiente per trattare una singola molecola

Per ulteriori dettagli si rimanda alla guida ufficiale di QE.

Si riassume il processo nello pseudo-codice che segue:

Algorithm 2 Schema di un algoritmo SCF per le equazioni K-S, base di onde piane

Inizializza la struttura cristallina, i parametri di reticolo e le posizioni atomiche
Genera una stima iniziale per $n(\vec{r})$
Scegli una griglia di campionamento k-point per la zona di Brillouin
Ciclo Principale:
while non convergente **do**
    Costruisci l'Hamiltoniana usando il potenziale efficace:
    $H_{KS} = T + v[n(\vec{r})]$
    Risolvi le equazioni K-S per ogni k-point:
    Risolvi $H_{KS}\phi_i(\vec{r}) = \epsilon_i\phi_i(\vec{r})$ per le funzioni d'onda ϕ_i e gli autovalori ϵ_i
    Aggiorna la densità elettronica:
    $n_{\text{new}}(\vec{r}) = \sum_{i=1}^{N} |\phi_i(\vec{r})|^2$
    Verifica la convergenza:
    if $|E[n_{\text{new}}(\vec{r})] - E[n(\vec{r})]| <$ `conv_thr` **then**
        convergente = True
    else
        $n(\vec{r}) =$ `mixing_beta` $\times\, n_{\text{new}}(\vec{r}) + (1 -$ `mixing_beta`$) \times n(\vec{r})$
    end if
end while
Output dei risultati: struttura delle bande, energia totale

L'*output* dell'eseguibile `pw.x` fornisce le energie delle bande di valenza, le forze sugli atomi, ed è anche possibile individuare la configurazione di equilibrio di una molecola.

Essendo interessati a calcolare l'*overlap* fra le funzioni d'onda di due configurazioni molecolari, è stato anzitutto scritto un programma, `school`, in grado di leggere le funzioni d'onda risultanti da `pw.x`, salvate in *file* con estensione `.xml` e `.wfc`.
Tale programma ha a sua volta necessità di un *file* di *input* in cui sono specificati:

- Il prefisso dei file `.xml` e `.wfc` da leggere
- Un parametro logico "`lsecfile`" che indica se leggere, o no, un secondo set di *file*

- Un numero "`y_displacement`" corrispondente alla traslazione dell'atomo di C lungo l'asse y rispetto alla configurazione di equilibrio, espresso in Bohr. Sarà rilevante nelle sezioni che seguono

Le funzioni d'onda di ciascuna banda, sotto forma di coefficienti $C_i(\vec{G})$ della serie di Fourier, sono salvate in un *array* bidimensionale di complessi, chiamato `evc`. Ha dimensione `npw` $\times$ `nbnd`, in cui `npw` è il numero di coefficienti.

Per effettuare le operazioni fra matrici si sfrutta la libreria `BLAS` per l'algebra lineare in Fortran, in particolare la *routine* `dgemm`. Si sottintende anche qui l'utilizzo di tale *routine* ogni volta che è presente un prodotto fra matrici.

Questa prima versione del programma `school` ha due scopi:

- Controlla l'ortonormalità delle funzioni d'onda calcolando l'*overlap* come prodotto dell'array `evc` con sé stesso. La condizione di ortonormalità è rispettata se il risultato è una matrice identità di dimensione `nbnd` $\times$ `nbnd`
- Calcola l'*overlap* fra le bande di due configurazioni della molecola. Il programma salva le funzioni d'onda in un altro *array* chiamato `evc_file`, dopo averle lette da una seconda serie di *file* `.xml`

Nota quindi la matrice di *overlap*, denominata `omat`, fra le bande di due diverse configurazioni della molecola, la si può passare all'algoritmo di minimizzazione.

Il codice per l'algoritmo è stato scritto in C come una funzione che ritorna `void` e ha come argomenti la matrice di *overlap* e il numero di bande. Il relativo *file* sorgente è stato aggiunto alla *directory* "`Modules`" di QE, ed è poi stata creata un'interfaccia alla funzione C nel codice "`school.f90`". La matrice `omat` viene passata all'algoritmo tramite un puntatore, `omat2`, al suo primo elemento.

Al termine, come descritto precedentemente, si ottiene la matrice che minimizza il funzionale $\mathcal{J}$ e che ha effetto di ruotare gli orbitali di valenza, portandoli da una configurazione all'altra. Tale matrice viene salvata, per comodità di programmazione, sovrascrivendo `omat`.
Per verificare il risultato dell'algoritmo, avendo a disposizione gli *array* per entrambe le configurazioni, si calcola:

$$\texttt{evc\_rot} := \texttt{omat} \times \texttt{evc}^{\mathsf{T}} \tag{17}$$

Poi la matrice di *overlap* viene nuovamente sovrascritta calcolando:

$$\texttt{omat} = \texttt{evc\_rot} \times \texttt{evc\_file} \tag{18}$$

Nelle relazioni sopra, `evc_rot` ha dimensione `nbnd` $\times$ `2*npw`. Il fattore 2 è dovuto al fatto che la *routine* `dgemm` è pensata per matrici reali, ma è più veloce rispetto alle funzioni per la moltiplicazione di matrici complesse: di conseguenza `evc` ed `evc_file` vengono passate a `dgemm` come matrici reali di dimensione `2*npw` $\times$ `nbnd`, e le dimensioni di `evc_rot` vengono raccordate di conseguenza.

Nello pseudo-codice 3 sono riassunti i passaggi.

Algorithm 3 Gestione e calcolo dell'overlap tra funzioni d'onda - `school`

1: Leggi le funzioni d'onda da *file* `.xml` e salva in `evc` ▷ Funzioni d'onda di una configurazione
2: Calcola `omat` = `evc` $\times$ `evc`$^{\mathsf{T}}$
3: **if** `lsecfile`=true **then**
4: Leggi le funzioni d'onda da *file* `.xml`, salva in `evc_file` ▷ Altra configurazione
5: Calcola `omat` = `evc` $\times$ `evc_file`$^{\mathsf{T}}$
6: **end if**
7: Chiama `algo(omat, nbnd)`
8: Calcola `evc_rot` = `omat` $\times$ `evc`$^{\mathsf{T}}$
9: Calcola `omat` = `evc_rot` $\times$ `evc_file` ▷ Overlap con orbitali ruotati

Esempio di applicazione: rotazione degli orbitali del CH_4

Le due configurazioni studiate per la molecola di metano sono mostrate in figura 1 e riportate in tabella 1. Queste coordinate sono state inserite in due file di *input* differenti, `methane_scf.in` e `methane_scf_displaced.in`, per poi eseguire `pw.x` con ciascuno.

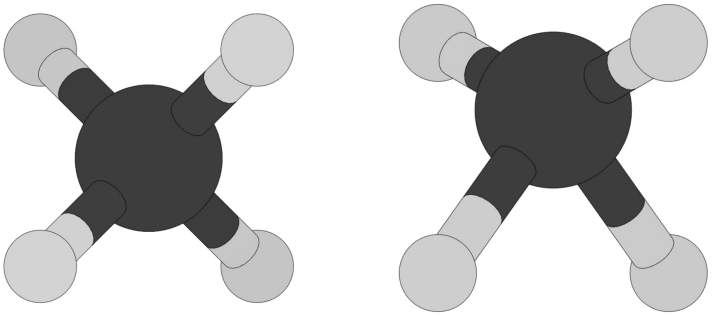

Figura 1: A sinistra la configurazione di equilibrio, a destra l'atomo di C è traslato di 0,5 Bohr

	Equilibrio (Bohr)		
	x	y	z
H	11.2	11.2	11.2
H	8.8	8.8	11.2
H	11.2	8.8	8.8
H	8.8	11.2	8.8
C	10.0	10.0	10.0

	C traslato (Bohr)		
	x	y	z
H	11.2	11.2	11.2
H	8.8	8.8	11.2
H	11.2	8.8	8.8
H	8.8	11.2	8.8
C	10.0	10.5	10.0

Tabella 1: Le coordinate cartesiane, in Bohr, delle due configurazioni

`ecutwfc` $= 40.0$	`nbnd` $= 5$
`mixing_beta` $= 0.5$	`conv_thr` $= 10^{-8}$

Tabella 2: Altri parametri di *input* per `PWscf` degni di nota

Per ciascuna delle due configurazioni viene avviato `pw.x` e poi `school.x`. Dal risultato di `check_g_real`, si nota che l'ortonormalità delle funzioni d'onda relative alle 5 bande è rispettata: la matrice di *overlap*, in entrambe le configurazioni, ha elementi sulla diagonale che si discostano da 1 di 10^{-15} e fuori diagonale lo scarto con zero è dell'ordine di 10^{-16}.

Viene poi chiamata `check_g_real_file` eseguendo di nuovo `school.x`, impostando `lsecfile=.true.` nel *file* di *input* `methane_school.in`. Il risultato è l'*overlap* fra le bande delle due configurazioni:

$$\texttt{omat} = \begin{pmatrix} -0.973 & 0.000 & 0.213 & 0.000 & 0.005 \\ -0.129 & 0.753 & -0.582 & 0.077 & 0.019 \\ 0.137 & 0.493 & 0.636 & 0.506 & -0.048 \\ -0.074 & -0.371 & -0.334 & 0.829 & 0.011 \\ -0.011 & 0.000 & 0.014 & 0.000 & -0.989 \end{pmatrix} \tag{19}$$

Notare che è stato ridotto il numero di cifre significative, come anche nei risultati che seguiranno.

A questa matrice di *overlap* corrisponde un valore iniziale per il funzionale *cost function* $\mathcal{J} \approx -0.26$, sfruttando l'equazione 10 e ricordando che si impone l'inizializzazione $U_0 = I$.

Passando `omat` all'algoritmo si ottiene convergenza ad un valore di $\mathcal{J} \approx -4,88$, come mostrato in figura 2. Facendo riferimento alla formula 13, si confronta la velocità per diversi valori del parametro μ, mentre in tutti i casi $T = 30$.
Il processo è stato replicato iterando fino a 1000 *step*, con $\mu = 0.1$, tuttavia $\mathcal{J}$ non diminuisce ulteriormente.

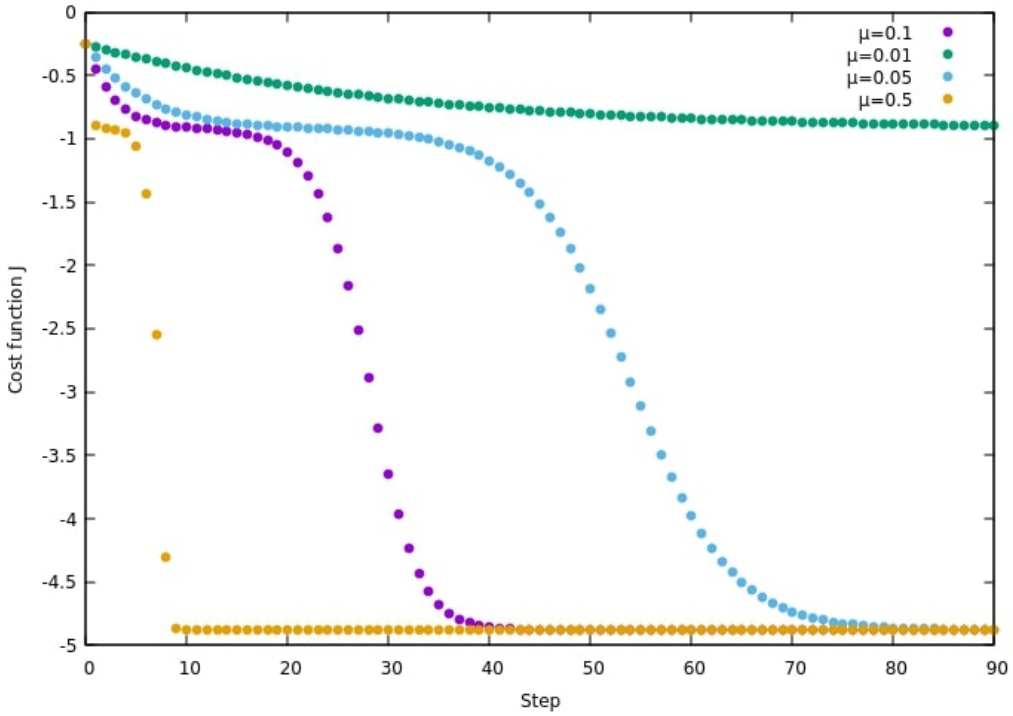

Figura 2: Convergenza della *cost function* al variare di μ

La stima della matrice di rotazione per gli orbitali è, al termine delle iterazioni:

$$\hat{U} = \begin{pmatrix} -0.977 & -0.001 & 0.214 & -0.001 & 0.009 \\ -0.135 & 0.773 & -0.615 & 0.079 & 0.006 \\ 0.146 & 0.507 & 0.672 & 0.519 & -0.020 \\ -0.078 & -0.381 & -0.353 & 0.851 & 0.004 \\ -0.013 & -0.007 & -0.017 & -0.007 & -1.000 \end{pmatrix} \tag{20}$$

Si verifica quindi l'equazione 15 calcolando:

$$\hat{U}O^{\mathsf{T}} = \begin{pmatrix} 0.996 & 0.001 & 0.001 & 0.001 & 0.005 \\ 0.001 & 0.963 & 0.012 & -0.006 & -0.013 \\ 0.001 & 0.012 & 0.961 & 0.007 & 0.028 \\ 0.001 & -0.006 & 0.007 & 0.971 & -0.008 \\ 0.005 & -0.013 & 0.028 & -0.008 & 0.988 \end{pmatrix} \tag{21}$$

Infine si copia $\hat{U}$ in `omat` e si seguono passaggi descritti nella sezione precedente, formule 17 e 18, ottenendo:

$$\texttt{omat} = \begin{pmatrix} 0.994 & 0.002 & -0.005 & 0.002 & -0.009 \\ -0.002 & 0.973 & 0.000 & 0.000 & -0.012 \\ -0.008 & 0.000 & 0.948 & 0.000 & -0.015 \\ -0.002 & 0.000 & 0.000 & 0.974 & -0.012 \\ 0.010 & 0.012 & -0.001 & 0.013 & 0.989 \end{pmatrix} \tag{22}$$

I risultati in 21 e 22 si discostano dalla matrice identità, nei termini peggiori, per un ordine di 10^{-2}. Entro questa soglia è quindi verificato che $\hat{U}$ corrisponda alla trasformazione unitaria che porta gli orbitali da una configurazione all'altra.

Esempio di applicazione: derivata degli orbitali del CH_4

Tramite la matrice di rotazione è possibile calcolare la derivata degli orbitali molecolari rispetto allo spostamento di un atomo, utile per il calcolo delle forze a cui sono soggetti gli atomi. Tale utilizzo sarà approfondito nella sezione seguente.
Come già illustrato, nel programma `school` vengono confrontate le funzioni d'onda di due configurazioni, la cui matrice di *overlap* viene passata all'algoritmo. Scegliendo particolari configurazioni si può stimare in buona approssimazione la derivata tramite il metodo delle differenze finite.
Consideriamo di nuovo la traslazione del C lungo l'asse y, di una quantità h. Chiamando ψ_+ e ψ_- le funzioni d'onda calcolate con C traslato, rispettivamente, di $+h$ e $-h$, si calcola la derivata come:

$$\frac{d\psi}{dy} = \frac{\psi_+ - \psi_-}{2h} + o(h^2) \tag{23}$$

Inoltre se gli orbitali, come verificato, sono ortonormali fra loro, vale che:

$$\langle \psi_i | \psi_j \rangle = \delta_{ij} \implies \left\langle \frac{d\psi_i}{dy} \middle| \psi_i \right\rangle = 0 \tag{24}$$

In cui $i, j = 1, ..., \texttt{nbnd}$.

Come descritto in precedenza, `evc` e `evc_file` sono gli *array* di coefficienti delle funzioni d'onda e in `omat` viene salvata la matrice di rotazione, mentre il conto `omat*evc=evc_rot` viene già effettuato nelle verifiche dell'algoritmo.
Si aggiorna quindi il programma per calcolare la differenza finita.
Con `y_displacement` si indica il parametro h, e viene specificato nel file di input per il programma `school.x`.
La formula 23 viene quindi implementata come:

$$\texttt{d\_psi\_dy} = \frac{\texttt{evc\_rot\_complex} - \texttt{evc\_file}}{2 * \texttt{y\_displacement}} \tag{25}$$

In cui `evc_rot_complex` è un *array* di complessi della dimensione corrispondente ad `evc_file` in cui è copiato il contenuto di `evc_rot`, che è invece un *array* di reali `nbnd` $\times$ `2*npw`.

Algorithm 4 Aggiunta a `school` per il calcolo della differenza finita

```
for j = 1 to nbnd do                          ▷ Copia evc_rot in un array di complessi evc_rot_complex
    for i = 1 to npw do
        evc_rot_complex(j, i) = cmplx(evc_rot(i,2*j-1),evc_rot(i,2*j))
    end for
end for
Calcola d_psi_dy = (evc_rot_complex − evc_file) / (2*y_displacement)   ▷ Array contenente la derivata
Inizializza scalar_product = 0
for i = 1 to nbnd do                          ▷ Prodotto scalare fra colonne
    scalar_product += dot_product(evc_file(:,i),d_psi_dy(:,i))
end for
Calcola omat = d_psi_d_y × evc_file           ▷ Matrice di overlap per la derivata
```

Si commentano ora i passaggi riassunti in algoritmo 4.
Notare che sono state inserite due funzioni specifiche di Fortran, il cui nome è esplicativo: `cmplx(,)` e `dot_product(,)`.
Per verificare la correttezza della derivata, viene calcolata la somma dei prodotti scalari fra le colonne della matrice `d_psi_d_y` e di `evc_file`, e salvata nella variabile `scalar_product`.
Un'ulteriore controllo è il calcolo dell'*overlap* fra queste due, il cui valore atteso è una matrice con zero sulla diagonale, come implicato in equazione 24. Tale matrice viene salvata in `omat`.
I risultati di tali controlli sulla derivata per alcuni valori del parametro di traslazione h sono riportati in tabella 3.

`y_displacement` (Bohr)	0.1	0.05	0.01
`scalar_product`	$-1.6 \cdot 10^{-2}$	$-5.9 \cdot 10^{-3}$	$-2.4 \cdot 10^{-4}$
`omat(1,1)`	$-1.7 \cdot 10^{-3}$	$-1.1 \cdot 10^{-3}$	$-3.5 \cdot 10^{-4}$
`omat(2,2)`	$-3.2 \cdot 10^{-3}$	$-3.3 \cdot 10^{-3}$	$3.3 \cdot 10^{-3}$
`omat(3,3)`	$5.0 \cdot 10^{-3}$	$5.3 \cdot 10^{-3}$	$5.0 \cdot 10^{-3}$
`omat(4,4)`	$-2.3 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$	$1.4 \cdot 10^{-3}$
`omat(5,5)`	$2.9 \cdot 10^{-3}$	$3.9 \cdot 10^{-3}$	$4.6 \cdot 10^{-3}$

Tabella 3: Confronto delle verifiche sulla derivata: prodotto fra bande e derivate, e *overlap*

Per tutti i casi sopra sono stati ripetuti anche i test sulla matrice di rotazione descritti nella sezione precedente. C'è da notare che il parametro di convergenza per `pw.x`, ossia `conv_thr`, è stato ridotto a 10^{-10} per il calcolo con $h = 0,05$ e a 10^{-12} per $h = 0,01$. Dato il valore iniziale della *Cost function* particolarmente svantaggioso, in questi due casi è stato necessario aumentare il numero di *step* di iterazione dell'algoritmo, come mostrato in figura 3.

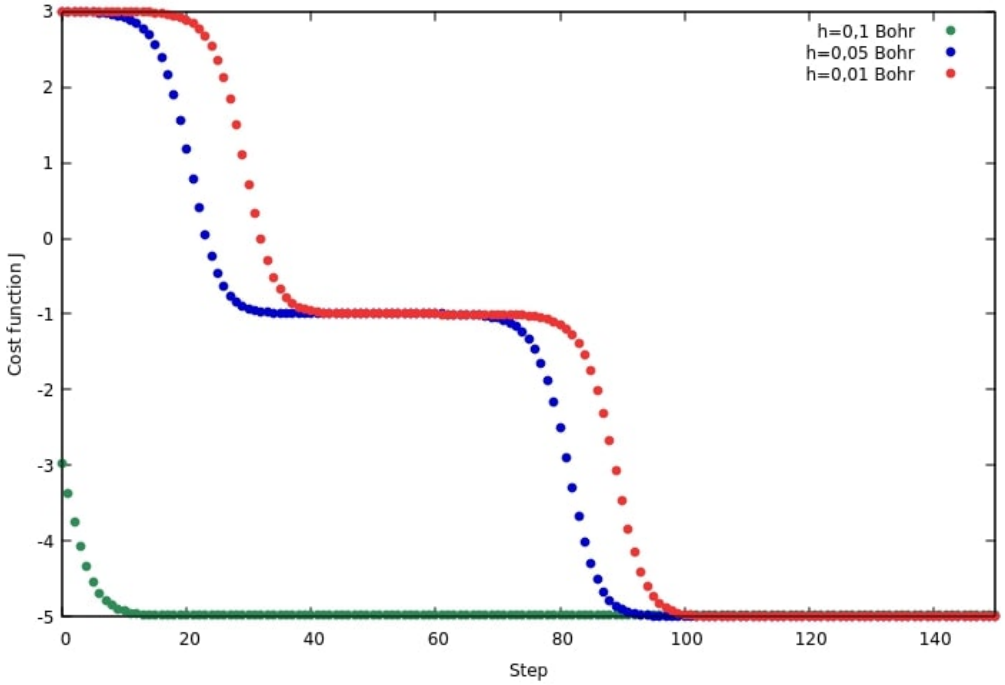

Figura 3: Convergenza della *cost function* per le diverse configurazioni, in tutti i casi $\mu = 0,1$

Le verifiche descritte in formule 21 e 22 effettuate per il caso in esame si riassumono in tabella 4

$h\,(Bohr)$	0.1	0.05	0.01
$\mathcal{J}$	$-5 + o(10^{-2})$	$-5 + o(10^{-3})$	$-5 + o(10^{-4})$
$\hat{U}O^{\mathsf{T}}$ (eq. 21)	$I \pm o(10^{-3} - 10^{-7})$	$I \pm o(10^{-3} - 10^{-7})$	$I \pm o(10^{-5} - 10^{-7})$
Overlap (eq. 22)	$I \pm o(10^{-2} - 10^{-11})$	$I \pm o(10^{-3} - 10^{-12})$	$I \pm o(10^{-3} - 10^{-12})$

Tabella 4: Risultati dell'algoritmo al variare del parametro di traslazione h

Applicazione: calcolo della forza in uno stato eccitato del CO

Consideriamo un sistema quantistico descritto da un'Hamiltoniana H che dipende da un parametro R, come la posizione di un nucleo. Se $\psi(R)$ è una funzione d'onda normalizzata che è autofunzione di $H(R)$ con energia $E(R)$ allora:

$$\langle\psi(R)|H(R)|\psi(R)\rangle = \langle E(R)\rangle \tag{26}$$

$$F = -\frac{d\langle E(R)\rangle}{dR} = -\frac{d}{dR}\langle\psi(R)|H(R)|\psi(R)\rangle \tag{27}$$

$$\frac{d\langle E(R)\rangle}{dR} = \langle\frac{d\psi}{dR}|H|\psi\rangle + \langle\psi|\frac{dH}{dR}|\psi\rangle + \langle\psi|H|\frac{d\psi}{dR}\rangle \tag{28}$$

Nel caso in cui $\psi(R)$ sia autofunzione dell'Hamiltoniana, $\frac{d\psi}{dR}$ è ortogonale a $\psi(R)$ e l'equazione si riduce al risultato del teorema di Hellmann-Feynman (H-F):

$$F = -\frac{d\langle E(R)\rangle}{dR} = -\langle\psi(R)|\frac{dH}{dR}|\psi(R)\rangle \tag{29}$$

La forza agente su un atomo può quindi essere generalmente espressa come l'opposto del gradiente dell'energia rispetto alla posizione $\vec{R}$ del nucleo.
L'energia, al termine del ciclo SCF, è ad un minimo rispetto alla densità $n(\vec{r})$. Lo spostamento infinitesimo di un nucleo modificherà il potenziale esterno v_{ext} e di conseguenza $n(\vec{r})$, tuttavia questa variazione non contribuisce alla derivata prima di E.
Chiamiamo ora il funzionale descritto in equazione 3 E_{KS}, e consideriamo un'altra forma del funzionale di energia, ossia il funzionale di H-K:

$$E_{HK}[n(\vec{r})] = E_{KS}[n(\vec{r})] + E_{II} \tag{30}$$

In cui E_{II} è l'energia potenziale elettrostatica fra i nuclei, non inclusa nell'Hamiltoniana di K-S. Nell'equazione per la forza rimarranno solo i termini relativi a v_{ext} e E_{II}, i quali sono gli unici a dipendere esplicitamente dalla posizione nucleare R:

$$F = -\int d\vec{r}\frac{dv_{ext}(\vec{r})}{dR} - \frac{dE_{II}}{dR} \tag{31}$$

Si rimanda al testo [2] per la trattazione teorica completa.

Nel caso in esame considereremo una configurazione eccitata della molecola di CO.

Le funzioni d'onda dello stato fondamentale, calcolate da `PWscf`, non saranno autofunzioni per H_{KS} relativa allo stato eccitato. La densità $n(\vec{r})$, da cui dipende H_{KS}, sarà infatti composta da un termine aggiuntivo relativo all'orbitale occupato:

$$n(\vec{r}) = \sum_i f_i\,\psi_i^*(\vec{r})\psi_i(\vec{r}) \tag{32}$$

In configurazione eccitata il fattore di occupazione f_i, che è 2 per orbitali occupati, sarà 1 per i due livelli più alti. Questi ultimi corrispondono all'HOMO (*Highest Occupied Molecular Orbital*, orbitale molecolare occupato più alto) e LUMO (*Lowest Unoccupied Molecular Orbital*, orbitale molecolare libero più basso) dello stato fondamentale.
In DFT, il primo teorema di H-K garantisce che la densità ottenuta dal ciclo SCF minimizza l'energia totale rispetto a variazioni della densità stessa. Tuttavia questo è vero solo per lo stato fondamentale. Ciò, oltre al fatto che non sono soddisfatte le ipotesi del teorema di H-F, implica che non è sufficiente sfruttare il risultato in equazione 31.

Si illustrano ora, qualitativamente, i passaggi per passare dallo stato fondamentale al primo eccitato:

- Si avvia `PWscf`, ottenendo le funzioni d'onda dello stato fondamentale
- A partire dall'*output* di `PWscf`, si popola il LUMO con un elettrone, allo stesso rimuovendolo dal HOMO
- Si ricalcolano densità, energia e forze nel nuovo stato

Questo è un processo che richiede la manipolazione dei numeri di occupazione, `wg`, e distribuire gli elettroni nei canali di spin, `nspin`. Vengono chiamate le routine `sum_band` ed una sua versione modificata per ricalcolare tutti i contributi all'energia del sistema, nonché `forces()`. Ulteriori dettagli tecnici andrebbero oltre lo scopo di questa tesi.

Nel computo delle forze mancano tuttavia dei termini, in quanto QE utilizza il risultato di H-F, ossia formule 29 e 31. Infatti, nei risultati stampati in *output*, le forze su ciascuno dei due nuclei non si bilanciano fra loro: alla forza sull'atomo la cui posizione è stata traslata in *input* mancano tali componenti. Questo fatto è riportato più avanti in tabelle 9, 10.

Le coordinate di *input* per `PWscf` sono riportate in tabella 5. In tabella 6 altri parametri notevoli.

I calcoli sono stati effettuati traslando sia il nucleo di O, sia il nucleo di C, con stesso parametro $h = 0.005$ Bohr lungo l'asse x.
`PWscf` è stato quindi avviato per 5 casi differenti.

	Coordinate (Bohr)		
	x	y	z
C	10.0	10.0	10.0
O	11.7	10.0	10.0

Tabella 5: Configurazione di default per le coordinate atomiche

`ecutwfc` $= 70,0$	`nbnd` $= 6$
`mixing_beta` $= 0.5$	`conv_thr` $= 10^{-10}$

Tabella 6

Il programma `school` è stato opportunamente modificato per effettuare le operazioni di manipolazione dei numeri di occupazione illustrate prima. Questo viene effettuato fornendogli i file `wfc` relativi alla configurazione di *default*.
Nello stesso programma, si effettuano poi i passaggi per il calcolo dei termini aggiuntivi della forza. Questi sono descritti nello pseudo-codice 5.

`school` viene avviato fornendogli i file `wfc` della configurazione di default, durante la esecuzione legge da altri `wfc` le funzioni d'onda per le configurazioni traslate.

Algorithm 5 Aggiornamento di `school`, derivata per stati eccitati

1: Carica le funzioni d'onda delle configurazioni 'zero', 'plus', 'minus'
2: Memorizza i coefficienti delle funzioni d'onda negli array `evc`, `evc_a`, `evc_b`
3: Chiama la routine `h_psi(evc)`, memorizza il risultato nell'array `hevc` ▷ Calcola $H_{KS}|\psi\rangle$
4: Calcola `omat_plus` = `evc_a`$^\mathsf{T}$ × `evc`
5: Calcola `omat_minus` = `evc_b`$^\mathsf{T}$ × `evc`
6: **for** $j = 1$ fino a nbnd **do** ▷ Riordina le colonne di `evc_a` per ottimizzare `omat_plus`
7: Trova `omat_plus(i,j)` con il valore assoluto più grande
8: **if** `omat_plus(i,j)` > 0 **then**
9: Scambia la colonna j con i di `evc_a`
10: Scambia gli elementi corrispondenti di `omat_plus`
11: **else**
12: Inverti il segno della colonna j in evc_rot_a
13: Inverti il segno della riga j in omat
14: **end if**
15: **end for**
16: Esegui la stessa ottimizzazione per `omat_minus`
17: Chiama `algo(omat_plus, nbnd)` ▷ Algoritmo SD
18: Chiama `algo(omat_minus, nbnd)`
19: Calcola `evc_rot_a` = `evc_a` × `omat_plus`
20: Calcola `evc_rot_b` = `evc_b` × `omat_minus`
21: Calcola `omat_plus` = `evc_rot_a`$^\mathsf{T}$ × `evc`
22: Calcola `omat_minus` = `evc_rot_b`$^\mathsf{T}$ × `evc`
23: Calcola `evc_dev` = $\frac{1}{2h}$(`evc_rot_a` − `evc_rot_b`) ▷ Differenza finita, secondo ordine
24: Calcola l'elemento di matrice aggiuntivo nell'equazione della forza
25: **for** i = 1 fino a `npw` **do**
26: **for** j = 1 fino a `nbnd` **do**
27: Calcola `force_dev` = `evc_dev(i,j)` × `hevc(i,j)` ▷ $\langle \frac{d\psi}{dR}|H_{KS}|\psi\rangle$
28: **end for**
29: **end for**

Anche in questo caso l'algoritmo di minimizzazione converge in modo soddisfacente.
Dopo aver applicato la rotazione, le matrici di *overlap* calcolate nei passaggi 21 e 22 di algoritmo 5 hanno elementi diagonali che si discostano da 1 di quantità da $o(10^{-4})$ a $o(-10^{-6})$.

Come confronto, la forza è stata stimata anche tramite la differenza finita dell'energia delle configurazioni:

$$F = \frac{E_+ - E_-}{2h} + o(h^2) \tag{33}$$

In cui E_+ ed E_- sono le energie totali delle due configurazioni eccitate, con uno dei due nuclei traslato di $+h$ e $-h$ rispettivamente.
Queste corrispondono al valore di aspettazione del funzionale E_{KS}, sono ottenute ricalcolando e sommando tutti i contributi in seguito alla popolazione del LUMO.

I valori ottenuti in *output* per le energie complessive sono riportati in tabella 7, mentre in tabella 8 vi sono gli autovalori di energia per ciascuna banda.

Si nota che, come prevedibile, variare quale dei due nuclei venga traslato non cambia significativamente l'energia complessiva: le differenze sono di $o(10^{-6})$.
Gli autovalori degli orbitali sono invece identici per entrambi i casi, entro la precisione di output di $o(10^{-15})$.

È importante notare i seguenti fatti:

- La differenza finita in formula 33 stima la forza totale

	E_+ (eV)	E_- (eV)
$x_O \pm 0.005$ (Bohr)	-41.99039	-41.99040
$x_C \pm 0.005$ (Bohr)	-41.97000	-41.96999

Tabella 7: Energie totali per ciascuna configurazione considerata

Numero di banda	Energia (eV)
1	-2.96
2	-1.38
3	-1.55
4	-1.55
5	-1.41
6	-0.14

Tabella 8: Livelli energetici per la configurazione eccitata, notare che 3 e 4 sono degeneri

- Il metodo descritto nei passaggi dell'algoritmo 5 è utilizzato per calcolare la derivata $\frac{d\psi}{dR}$, quindi i termini "misti" nell'espressione della forza 28
- In entrambi i metodi il risultato corrisponde alla forza, o la sua componente, agente sul nucleo che viene traslato nell'*input* di `PWscf`

In tabelle 9 e 10 sono riportati i risultati relativi al calcolo delle forze.

C, termine $\langle\psi\|\frac{dH}{dR}\|\psi\rangle$	-1.80
O, termine $\langle\psi\|\frac{dH}{dR}\|\psi\rangle$ (a)	3.71
O, termine $\langle\frac{d\psi}{dR}\|H\|\psi\rangle + \langle\psi\|H\|\frac{d\psi}{dR}\rangle$ (b)	-1.60
Forza complessiva su O (a+b)	2.11
Forza su O, da formula 33	2.04

Tabella 9: Componenti della forza ($Ry/Bohr$) riferiti a ciascun nucleo, caso di O traslato

C, termine $\langle\psi\|\frac{dH}{dR}\|\psi\rangle$ (a)	-1.80
O, termine $\langle\psi\|\frac{dH}{dR}\|\psi\rangle$	3.71
C, termine $\langle\frac{d\psi}{dR}\|H\|\psi\rangle + \langle\psi\|H\|\frac{d\psi}{dR}\rangle$ (b)	-0.12
Forza complessiva su C (a+b)	-1.92
Forza su C, da formula 33	-2.04

Tabella 10: Componenti della forza ($Ry/Bohr$) riferiti a ciascun nucleo, caso di C traslato

Conclusioni

Avendo applicato l'algorimo in tre contesti differenti, se ne commentano i risultati.

Nell'applicazione sul CH_4, i risultati seguenti la rotazione e il calcolo numerico delle derivate sono soddisfacenti.
Riguardo quest'ultimo, il prodotto tra derivate e orbitali (tabella 3) si discosta da zero di una quantità corrispondente all'errore intrinseco del metodo della differenza finita, cioè h^2.

Nel caso della molecola di CO, dall'output di `PWscf` si nota che alcuni orbitali sono degeneri.
Gli orbitali degeneri sono intrinsecamente ambigui quando vengono messi in relazione tra diverse configurazioni, e valgono le seguenti considerazioni:

- La matrice di rotazione può aiutare ad allineare gli orbitali degeneri tra le configurazioni, e quindi a calcolare le derivate accuratamente
- Gli orbitali non degeneri hanno un chiaro corrispondente tra le configurazioni, rendendo il loro trattamento più semplice. Per questi, infatti, la differenza finita può essere calcolata direttamente

Per affinare ulteriormente l'approccio, l'algoritmo potrebbe essere applicato solo agli orbitali degeneri, riducendo costo computazionale ed evitando di disturbare la corrispondenza per i non degeneri.

Includere più bande (`nbnd`) di quelle necessarie può introdurre rumore numerico e complicare il processo di rotazione e di ottimizzazione della corrispondenza fra configurazioni.
Ciò avviene perché livelli ad energia superiore possono avere termini di *overlap* con orbitali più bassi non trascurabili, e questo può portare l'algoritmo a risultati peggiori.

La molecola di CO, essendo eteronucleare, ha asimmetrie elettroniche intrinseche: spostare il C rispetto all'O può avere effetti diversi sulla struttura elettronica. Questo spiega i risultati del calcolo delle forze, la cui differenza, per i due casi, è maggiore del previsto.

Bibliografia

[1] Traian E Abrudan, Jan Eriksson, and Visa Koivunen. Steepest descent algorithms for optimization under unitary matrix constraint. *IEEE Transactions on Signal Processing*, 56(3):1134–1147, 2008.

[2] Richard M Martin. *Electronic structure: basic theory and practical methods*. Cambridge university press, 2020.

[3] Patrick Sit and Linghai Zhang. *Density Functional Theory in Heterogeneous Catalysis*, chapter 23, pages 405–418. John Wiley & Sons, Ltd, 2021.