Dipartimento di Fisica
Corso di Laurea Triennale

Studio della dissociazione del catione idrogenonio (H_3^+) attraverso l'uso di computer quantistici

Relatore:
Dr. Andrea Giachero

Correlatore:
Stefano Barison

Candidato:
Rodolfo Carobene

Matricola:
838092

ANNO ACCADEMICO 2020/2021

Sommario

Questo elaborato presenta lo studio dell'energia di dissociazione del catione idrogenonio (formula chimica: H_3^+) tramite l'uso di metodi di computazione quantistica.

Inizieremo introducendo l'idea di computer quantistico, dalla sua presentazione ai più recenti sviluppi. Nel primo capitolo sarà poi presentato il problema elettronico nel caso di sistemi molecolari. Un primo approccio alla risoluzione di questo problema sarà fornito attraverso i metodi computazionali classici della chimica quantistica. Tali metodi saranno infine analizzati e confrontati.

Nel secondo capitolo verranno introdotti gli elementi fondamentali della computazione quantistica nonché l'algoritmo utilizzato per il calcolo delle energie molecolari: il *Variational Quantum Eigensolver* (VQE). Verranno inoltre illustrati alcuni problemi con cui è necessario confrontarsi nell'uso dei computer quantistici attuali, quali il rumore e la decoerenza, nonché alcune possibili soluzioni sia dal punto di vista *software*, attualmente implementabili, sia da quello *hardware*, che potranno essere utilizzate in futuro. Infine, verranno introdotti i principali strumenti necessari alla scrittura e all'esecuzione di programmi su computer quantistici: Qiskit, *framework* di programmazione sviluppato da IBM, e IBMQ, progetto che rende accessibili a chiunque attraverso il cloud alcuni computer quantistici.

Nel penultimo capitolo verrà affrontato il problema pratico dello studio dell'energia di dissociazione di H_3^+. Inizialmente si utilizzeranno i metodi classici introdotti al capitolo 2, al fine di avere dei risultati con i quali confrontarsi. Successivamente, verranno svolte diverse simulazioni di VQE. In particolare verrà svolta una prima simulazione in assenza di rumore, per poi passare all'introduzione di un modello di rumore importato direttamente da un quantum computer *reale*. A questo punto, sarà necessario affrontare il problema dell'eccessiva distorsione dei risultati. Per questo verranno svolte altre simulazioni con strumenti appositi per ridurre al minimo il rumore.
Nell'ultima sezione di questo capitolo è descritta l'esecuzione dell'algoritmo su un computer quantistico vero e proprio e ne sono analizzati i risultati.

Infine, nel capitolo conclusivo saranno riassunti i risultati teorici e sperimentali dei capitoli precedenti. A questi saranno aggiunte considerazioni su possibili sviluppi futuri della ricerca, volti a migliorare le performance dell'algoritmo.

Indice

Elenco delle figure

Introduzione

Un computer è una macchina automatizzata programmabile in grado di svolgere operazioni matematiche e logiche e di memorizzare informazioni a velocità e in quantità superiori a quelle di cui è comunemente capace il cervello umano[1]. Il principio di fondo dell'informatica classica è che ogni possibile problema computazionale può essere espresso tramite operazioni logiche consecutive, tuttavia il fatto che ogni problema sia risolvibile da un computer non significa certo che lo sia in maniera efficiente. Questa proprietà va a nostro vantaggio nella crittografia, ad esempio: quando criptiamo un file non stiamo supponendo che sia impossibile trovare la nostra chiave (questo sarebbe un problema computazionale facilmente esprimibile sotto forma di programma); bensì stiamo confidando nel fatto che sia impossibile trovarla in un tempo ragionevole.
Ovviamente la questione del tempo necessario a rispondere ad alcun quesiti non è legata unicamente alla crittografia, ma coinvolge una moltitudine di particolari problemi o sistemi che non siamo in grado di risolvere in maniera efficiente. Ad esempio è estremamente difficile, per un computer classico, simulare un arbitrario sistema quantistico, poiché il costo computazionale di un problema di tal genere scala in modo esponenziale con la dimensione dell'input (ad esempio con il numero di elettroni considerati in una molecola).
In questo contesto si può inquadrare l'importanza dell'introduzione dei computer quantistici che sono (o saranno) in grado di oltrepassare i limiti della computazione classica[1]. L'idea di un processore quantistico è stata proposta per la prima volta nel 1982 da Richard Feynman[2] come nozione puramente astratta. Secondo la sua definizione, poi approfondita da Di Vincenzo[3], un computer quantistico è un sistema che *esegue* (dunque non si limita a una simulazione) un problema regolato dalle leggi della meccanica quantistica. È una definizione molto ampia ed ha dato luogo, nella breve storia della computazione quantistica, a diverse implementazioni: è molto interessante sottolineare che anche adesso esistono computer quantistici basati su tecnologia differenti, come si può vedere in Fig. 0.0.1. I primi computer quantistici, in ogni caso, sono stati realizzati con successo solo negli ultimi anni dello scorso millennio, ma da allora si sono estremamente evoluti: basti pensare che recentemente è stato proposto un "quantum computer da scrivania"[4][2]. Un'innovazione che ricalca, in qualche modo, l'introduzione dei primi *personal computer* negli anni '80.

[1] definizione da *Oxford Languages*.
[2] Discover: A Desktop Quantum Computer for Just 5000$

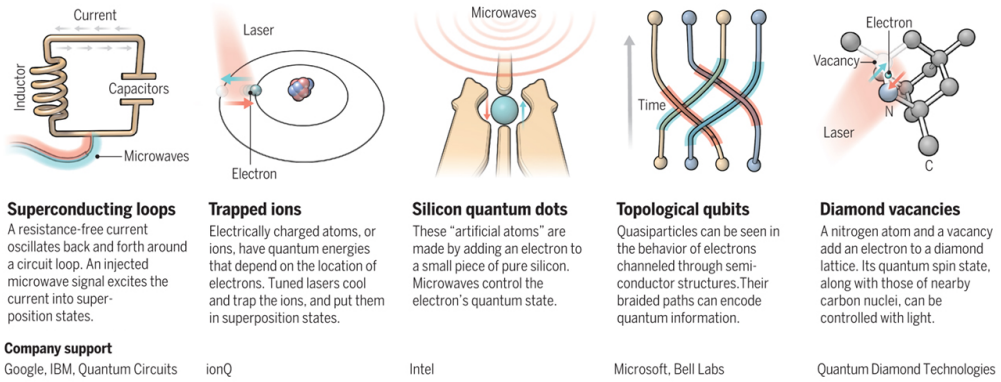


Figura 0.0.1: Esempi di sistemi fisici attraverso i quali si può costruire un quantum computer

immagine di Noteworthy - The Journal Blog

In una prima fase della ricerca sulla computazione quantistica ci si é concentrati sulla realizzazione di algoritmi che permettevano di risolvere in modo efficiente problemi difficili da affrontare classicamente. La progettazione di un algoritmo di ricerca (algoritmo di Grover[5]) e di fattorizzazione (algoritmo di Shor[6]) hanno mostrato il decisivo vantaggio dei processori quantistici su quelli classici per alcuni problemi specifici.
Entrambi gli algoritmi, tuttavia, presuppongono processori che obbediscono perfettamente alle istruzioni che vengono a loro impartite e presuppongono, inoltre, di avere computer senza alcun limite di potenza computazionale, ma attualmente entrambi questi requisiti non sono soddisfatti e non è possibile utilizzare tali algoritmi per problemi di pratica utilità.
Quella in cui siamo viene definita l'era dei NISQ: *Noisy Intermediate-Scale Quantum computers/algorithms/computing*[7] (si suppone che terminerà in due o tre anni, come visibile in figura 0.0.2).
I computer NISQ hanno un ridotto numero di qubit (l'unità fondamentale d'informazione) e sono soggetti a varie forme di rumore che bisogna tenere in considerazione nella progettazione di algoritmi.
In questa categoria ricade VQE (*Variational Quantum Eigensolver*) su cui il lavoro di questa tesi si concentra. Il VQE, come molti altri algoritmi per computer quantistici, non tenta di raggiungere una soluzione esatta a meno di una certa tolleranza, ma tenta di raggiungere una buona soluzione approssimata.
Il VQE è stato introdotto per la prima volta da A. Peruzzo (2014)[8] ed è un algoritmo che si occupa di trovare l'energia di stato fondamentale per una determinata molecola. Essendo stato introdotto così recentemente, sono ancora in corso numerosi studi per migliorarne l'efficienza e per superare alcune difficoltà che si incontrano nel suo utilizzo e di cui parleremo nei capitoli successivi.
L'obiettivo principale di questa tesi sarà lo studio dell'energia di dissociazione del catione idrogenonio H_3^+ il quale, per alcune sue caratteristiche, si presta particolarmente a uno studio energetico tramite VQE.

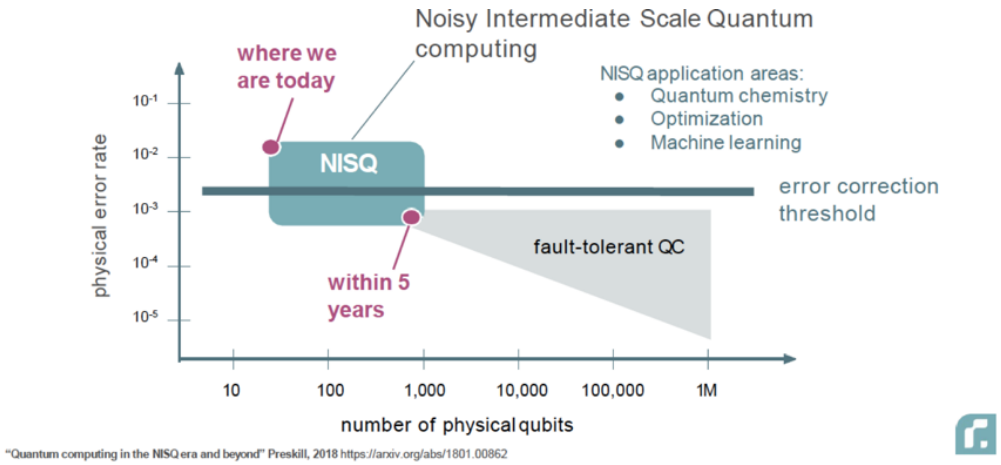


Figura 0.0.2: Evoluzione prevista dei quantum computer (2018)
immagine di towards data science

A questo fine, sarà necessario introdurre i fondamenti teorici per comprendere tutti gli aspetti dell'implementazione del VQE[9]: a partire dall'analisi degli strumenti della chimica computazionale classica fino ad arrivare alle necessarie conoscenze di computazione quantistica.

Capitolo 1

Analisi del problema e soluzioni classiche

In questo capitolo introdurremo il problema affrontato in questa tesi e illustreremo alcune soluzioni computazionali tradizionali e i loro limiti. Per un approfondimento sui temi di questo capitolo si veda [10].

1.1 Problema elettronico

Un sistema molecolare composto da molteplici atomi necessita, per essere risolto, delle leggi della meccanica quantistica; in questo contesto risolvere significa trovare i livelli energetici, ovvero gli autovalori dell'operatore hamiltoniano del sistema, e le autofunzioni ad essi relative. Il problema, espresso tramite l'equazione di Schrödinger indipendente dal tempo, risulta:

$$\hat{H}\left|\psi\right\rangle = (\hat{T}_N + U_{N-N} + \hat{T}_e + U_{e-N} + U_{e-e})\left|\psi\right\rangle = E\left|\psi\right\rangle \tag{1.1.0.1}$$

Considerando (i,j) indici che variano su tutti i nuclei e (k,l) sugli elettroni, abbiamo:

- $\hat{T}_N$: operatore energia cinetica dei nuclei;

$$\hat{T}_N = \sum_i -\frac{\hbar^2}{2m_i}\nabla_i^2 \tag{1.1.0.2}$$

- U_{N-N} : energia potenziale di interazione nucleo-nucleo;

$$U_{N-N} = \frac{1}{2}\sum_{i,j}\frac{e_0^2 Z_i Z_J}{4\pi\epsilon_0\left|R_i - R_j\right|} \tag{1.1.0.3}$$

- $\hat{T}_e$: operatore energia cinetica degli elettroni;

$$\hat{T}_e = \sum_k -\frac{\hbar^2}{2m_i}\nabla^2 k \tag{1.1.0.4}$$

- U_{e-N} : energia potenziale di interazione elettrone-nucleo;

$$U_{e-N} = \sum_{i,k} \frac{e_0^2 Z_i}{4\pi\epsilon_0 |R_i - x_k|} \tag{1.1.0.5}$$

- U_{e-e} : energia potenziale di interazione elettrone-elettrone.

$$U_{e-e} = \frac{1}{2} \sum_{k,l} \frac{e_0^2}{4\pi\epsilon_0 |x_k - x_l|} \tag{1.1.0.6}$$

Il problema è troppo complicato per essere risolto in modo analitico e dobbiamo necessariamente affidarci a varie approssimazioni. Per prima cosa separiamo il termine di energia cinetica nucleare dalla parte elettronica, che contiene il termine elettronico cinetico e tutti i termini potenziali, dipendente solo parametricamente dalla posizione dei nuclei.
Il problema elettronico risulta:

$$\hat{H}_{el} |\psi_{el}\rangle = (\hat{T}_e + U_{e-N} + U_{e-e} + U_{N-N}) |\psi_{el}\rangle = E_{el} |\psi_{el}\rangle \tag{1.1.0.7}$$

Si può dimostrare che gli autostati ψ_{el} formano un sistema ortonormale completo, perciò possiamo esprimere le autofunzioni del problema complessivo come:

$$\psi = \sum_i \chi(R_i)\psi_{el}(R_i) \tag{1.1.0.8}$$

I coefficienti χ dipendono anch'essi parametricamente, come le autofunzioni elettroniche, dalle posizioni fissate dei nuclei.
Il procedimento di risoluzione generale, dunque, consiste nel risolvere il problema elettronico per varie distanze interatomiche e solo a questo punto nell'introdurre l'operatore di energia cinetica dei nuclei e con esso l'energia vibrazionale, rotazionale e traslazionale della molecola[1].

1.1.1 Approssimazione di Born-Oppenheimer

L'approssimazione di Born-Oppenheimer, o approssimazione adiabatica, consiste nel considerare completamente indipendenti il problema nucleare da quello elettronico. In particolare è possibile trascurare la sommatoria nell'equazione 1.1.0.8 e analizzare, quindi, un unico stato elettronico "alla volta": la massa dei nuclei, infatti, è sensibilmente maggiore di quella elettroni e di conseguenza i tempi caratteristici del moto dei nuclei stessi non sono comprabili a quelli elettronici. Sotto queste condizioni abbiamo la funzione d'onda complessiva scrivibile come:

$$\psi = \chi(R_i)\psi_{el}(R_i) \tag{1.1.1.1}$$

[1]Si pone l'attenzione sul fatto che l'energia cinetica nucleare non sia mai nulla (in particolare non lo è l'energia vibrazionale che ha un termine di punto zero), cionondimeno in questa tesi mi occuperò unicamente del problema elettronico.

Inserendo questa soluzione nel problema complessivo (equazione 1.1.0.1) si ottiene:

$$(\hat{T}_N + \hat{H}_{el}) \left|\chi(R_i)\psi_{el}(R_i)\right\rangle = (\hat{T}_N + E_{el}) \left|\chi(R_i)\psi_{el}(R_i)\right\rangle \tag{1.1.1.2}$$

Il secondo elemento di approssimazione adiabatica è legato al trascurare, nell'equazione appena scritta, il termine $\hat{T}_N \left|\psi_{el}\right\rangle$. A questo punto è sufficiente studiare il problema nucleare con operatore hamiltoniano cinetico e un termine di potenziale efficacie dato dalla soluzione del problema elettronico:

$$(\hat{T}_N + \hat{H}_{el}) \left|\chi(\{R_i\})\right\rangle \tag{1.1.1.3}$$

Riassumendo, per quel che concerne questa tesi, l'approssimazione di Born-Oppenheimer ci consente di studiare in modo completamente separato il problema elettronico e quello nucleare.

1.1.2 Basi per le funzioni d'onda

Una funzione d'onda, per un qualsiasi sistema di elettroni, deve essere completamente antisimmetrica, per rispettare il principio di esclusione di Pauli. Questo si risolve nello scrivere la funzione d'onda come somma di determinanti di Slater di funzioni di singola particella (dove x_i si riferisce alla posizione dell'i-esima particella, il pedice $_{A,B,...}$ si riferisce agli autovalori del CSCO[2] che descrivono un singolo stato e infine N è il numero complessivo di orbitali di spin):

$$\Psi(x_i, ..., x_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_A(x_1) & \psi_A(x_2) & ... \\ \psi_B(x_1) & \psi_B(x_2) & ... \\ ... & ... & ... \end{vmatrix} \tag{1.1.2.1}$$

La funzione Ψ appartiene allo spazio di Hilbert $\mathscr{H} = L^2(\mathbb{R}^3) \times \mathbb{C}^2$ e può essere espressa come composizione lineare di una qualsiasi base per tale spazio che, tuttavia, è un infinito dimensionale. Questo ci porta a definire una base dimensionalmente finita che, in quanto tale, non può che portare a una funzione d'onda approssimata. Sarà dunque fondamentale la scelta di una base che meglio riesca ad approssimare la funzione Ψ pur mantenendo sotto controllo la complessità computazionale [11].

Funzioni di Slater (STO)

In chimica computazionale un primo tentativo si può fare con le funzioni di Slater (*Slater Type Orbitals*, identificate con la sigla STO):

$$\phi^{STO}_{a,b,c,\zeta}(\mathbf{r}) = v_{abc} x^a y^b z^c e^{-\zeta|\mathbf{r}|} \tag{1.1.2.2}$$

Le funzioni STO formano serie rapidamente convergenti e sono un'ottima approssimazione a corto e a lungo raggio, tuttavia gli integrali che entrano in gioco calcolando gli autovalori dell'hamiltoniano sono complessi e richiedono un grande quantitativo di tempo (soprattutto per molecole poliatomiche dove gli orbitali hanno centri differenti).

[2] i.e. *Complete Set of Commuting Operators*

Funzioni Gaussiane (GTO)

Il passo successivo, per superare i problemi delle STO, è rappresentato dalle GTO (*Gaussian Type Orbitals*) o funzioni gaussiane:

$$\phi^{GTO}_{a,b,c,\zeta}(\mathbf{r}) = v_{abc} x^a y^b z^c e^{-\zeta|\mathbf{r}|^2} \tag{1.1.2.3}$$

Le funzioni GTO portano a dei calcoli estremamente semplici e rapidi sebbene, per contro, convergano più lentamente e per questo sia necessario utilizzarne in numero maggiore affinché portino a un'approssimazione accettabile.

Funzioni Gaussiane Contratte (CGTO)

Per combinare i pregi di STO e GTO è possibile utilizzare un certo numero di funzioni gaussiane per approssimare una singola STO:

$$\phi^{CGTO}_{a,b,c,\zeta}(\mathbf{r}) = v_{abc} x^a y^b z^c \sum_{i=1}^{N} e^{-\zeta_i|\mathbf{r}|^2} \tag{1.1.2.4}$$

Quest'ultimo modello è quello delle funzioni gaussiane contratte.

Basi Minimali (STO-nG)

I più semplici set di base utilizzati in chimica computazionale vengono chiamati STO-nG (malgrado siano particolari CGTO). In questa rappresentazione è utilizzata un'unica funzione STO per ciascun orbitale di spin[3]. La "n" nel nome identificativo della base è un indice variabile che indica il numero di funzioni gaussiane utilizzate per approssimare una funzione di Slater.
Le basi minimali sono molto comode per la loro semplicità, ma sono spesso inaccurate. Si può cercare di correggere queste limitazioni in vari modi: aumentando il numero di GTO per gli orbitali (talvolta solo per quelli più esterni) o aggiungendo particolari funzioni (di polarizzazione, indicate con un asterisco nel simbolo della base, o funzioni diffuse, identificate da un +) per studiare effetti che la base minimale non riesce a considerare.

Basi di Pople (l-mnG)

Per correggere alcuni difetti delle basi minimali si possono considerare un maggior numero di gaussiane per ogni orbitale: questo è il fondamento delle basi di Pople indicate con l-mnG. Questo tipo di basi considera un numero differente di GTO per ciascun orbitale atomico a seconda che esso sia di valenza o meno (si parla di basi di *split-valence*); in particolare i numeri nel nome della specifica base indicano:

- l: numero di GTO per approssimare gli orbitali più interni;

[3]Gli orbitali da considerare dipendono dalla variante del metodo di Hartree-Fock utilizzata, questo argomento viene trattato più avanti.

- m,n: numeri indicanti il numero di GTO utilizzate per approssimare gli orbitali di valenza. Il fatto che siano due (si può aumentarne il numero) indica che ogni orbitale esterno è combinazione lineare di due diversi orbitali: il primo composizione di m GTO e il secondo composizione di n GTO. In genere uno di questi orbitali è utilizzato per rappresentare un orbitale più localizzato intorno al nucleo mentre l'altro per rappresentare un orbitale più diffuso.

Basi correlate

Le *correlation-consistent bases* sono basi di funzioni più complesse e di vario tipo, ottimizzate con tecniche semi-empiriche per calcoli successivi all'approssimazione di Hartree - Fock, della quale parleremo nella successiva sezione.
Le basi correlate polarizzate vengono indicate con cc-pVXZ. L'unico indice variabile è "X" (D,T,Q,5...) che rappresenta la molteplicità di GTO usata per rappresentare gli orbitali di valenza (la V di cc-pVXZ) sta appunto per "valence".
Se al nome della base viene aggiunto il prefisso "aug" si sta indicando l'aggiunta di un set di funzioni diffuse per ciascun momento angolare della base (evidentemente dipendente dalla molecola).
Altre tipologie di basi correlate sono presenti, ad esempio per tenere conto degli effetti relativistici, ma non verranno considerate in questa tesi.

1.1.3 Approssimazione di Hartree-Fock

Il metodo di Hartree-Fock è utilizzato per risolvere il problema elettronico in atomi e molecole in modo approssimato. Le posizioni dei nuclei vengono fissate e viene applicata anche l'approssimazione di Born-Oppenheimer.
L'idea fondamentale è che gli elettroni occupino dei precisi orbitali in particolari configurazioni antisimmetriche rapresentate da funzioni d'onda. Tale congettura è un'approssimazione giustificata dal principio variazionale che ci permette di andare a variare le funzioni col solo scopo di minimizzare l'energia risultante per avvicinarci allo stato fondamentale.
Il metodo di Hartree-Fock ha tre fondamentali varianti:

- *Restricted Hartree-Fock* (RHF), dove ogni orbitale spaziale è occupato doppiamente;
- *Open-shell Restricted Hartree-Fock* (ROHF), dove non tutti gli orbitali sono doppiamente occupati (benchè sia possibile occuparli parzialmente con elettroni in spin up o down indifferentemente);
- *Unrestricted Hartree-Fock* (UHF), dove elettroni con spin up e down occupano (o possono occupare) orbitali differenti.

La scelta del metodo da utilizzare è dettata dalla molecola che si vuole studiare: ad esempio, con H_3^+ ci aspettiamo che i due elettroni occupino un singolo orbitale molecolare, permettendoci di usare il metodo RHF.

Complessivamente, l'hamiltoniano di singola particella è dato dalla parte cinetica, dall'interazione elettrone-nucleo e dalla somma di due potenziali di campo medio: il potenziale di Hartree (che considera il campo elettrostatico classico generato dagli altri elettroni) e il potenziale di Fock (che è un corrispettivo quantistico legato alla degenerazione di scambio).
Il metodo HF consente di risolvere il problema agli autovalori dell'equazione di Schrödinger in modo autoconsistente: scelta una base, si calcolano i potenziali di Hartree-Fock a partire da un primo ansatz[4], si risolve l'equazione di Schrödinger e si ricalcolano i potenziali con le nuove autofunzioni (il tutto reiterato fino a convergenza).
Per un singolo elettrone:

$$V_H(\bar{x})\psi_i = \frac{e_0^2}{4\pi\epsilon_0}\int d^3x \sum_j \frac{\psi_j^*(x')\psi_j(x)}{\left|\bar{x}-\bar{x'}\right|}\psi_i(x) \tag{1.1.3.1}$$

$$V_F(\bar{x})\psi_i = \frac{e_0^2}{4\pi\epsilon_0}\int d^3x' \sum_j \frac{\psi_j^*(x')\psi_j(x)}{\left|\bar{x}-\bar{x'}\right|}\psi_i(x') \tag{1.1.3.2}$$

Perchè un computer possa risolvere efficientemente questo problema bisogna scrivere un'equazione matriciale agli autovalori, detta equazione di Hartree-Fock-Roothan (HFR):

$$FC = SC\epsilon \tag{1.1.3.3}$$

Dove la matrice F è costruita a partire dall'operatore di Fock applicato fra singole funzioni:

$$F_{a,b} = \left(\int dx_2 \frac{\psi_b^*(x_2)\psi_a(x_2)}{|x_1-x_2|}\right)\psi_b(x_1) \tag{1.1.3.4}$$

La matrice S è la matrice di overlap:

$$S_{a,b} = \int dx\, \phi_a^*(x)\phi_b(x) \tag{1.1.3.5}$$

E, infine, la matrice C contiene i vettori dei coefficienti dell'espansione della funzione d'onda in considerazione sulla base scelta:

$$\psi_a = \sum_b C_{ab}\phi_b \tag{1.1.3.6}$$

Come già detto, dato un ansatz, possiamo generalmente risolvere il problema in modo autoconsistente.

[4] i.e. Una prima ipotesi.

1.2 Oltre il campo medio: metodi computazionali classici

L'approssimazione di campo medio di Hartree-Fock porta a un errore nel risultato finale che può essere addirittura $\sim 1\,eV$ poiché essa non considera il fenomeno dell' *electron correlation*, ovvero il fatto che il moto di un elettrone risulta correlato con quello degli altri elettroni[5]. Evidentemente, per energie di stato fondamentale dell'ordine della decina di elettronvolt, questo errore non è affatto trascurabile e bisogna, dunque, utilizzare dei metodi che migliorino il risultato dell'approssimazione di Hartree-Fock[12].

1.2.1 Teoria perturbativa di Møller-Plesset (MP)

Una prima soluzione si può ricavare introducendo l'interazione elettrone-elettrone come perturbazione.
Possiamo scrivere come $\hat{H}_0$ l'hamiltoniano ricavato col metodo di Hartree-Fock e introdurre la perturbazione $\hat{V}$ variabile con un parametro λ:

$$\hat{H} = \hat{H}_0 + \lambda\hat{V} \tag{1.2.1.1}$$

Questo metodo prende il nome di correzione perturbativa di Møller-Plesset e porta a soluzioni migliori di HF senza troppe complicazioni teoriche.
La perturbazione $\hat{V}$ contiene operatori di doppia eccitazione[6] che modellizzano la promozione di coppie di elettroni da orbitali occupati (quelli ottenuti da HF) a orbitali liberi (in qualche modo così si dà più spazio agli elettroni per stare fra di loro lontano).
Il problema principale della MP è il rapido aumento di complessità con il numero di elettroni ed orbitali (e dunque all'aumentare della complessità della base) poiché in $\hat{V}$ devono essere considerate tutte le possibili doppie eccitazioni. Inoltre, partendo da HF, ne eredita alcuni problemi: ad esempio, HF spesso sovrastima le barriere di potenziale e anche MP è caratterizzato dallo stesso errore.
Vi sono anche delle versioni leggermente modificate di MP che cercano di risolvere i problemi intrinseci del metodo e dunque, malgrado tutto, l'approccio perturbativo rimane valido in virtù, soprattutto, della sua semplicità teorica.

1.2.2 Full Configuration Interaction (FCI)

Un altro metodo per considerare l'energia correlata è quello della *Configuration Interaction* (CI). Anche CI, così come MPTT, si basa sugli operatori di eccitazione e diseccitazione, ma non utilizza un approccio di tipo perturbativo, bensì un approccio esatto a partire dalla funzione d'onda.
Partiamo dal presupposto, dunque, di aver già applicato il metodo di Hartree-Fock giungendo a una funzione d'onda $|\psi_{HF}\rangle$. Per considerare tutti i possibili altri stati

[5] Quando in sezione 1.1.2 abbiamo parlato di basi correlate per calcoli successivi ad Hartree Fock ci riferivamo proprio alle correzioni di tale fenomeno.

[6] MP può essere usato a più livelli di complessità considerando solo eccitazioni singole (MP1), eccitazioni singole e doppie (MP2) e così via. MP2, che ho qui considerato, è lo standard nonché il primo livello di miglioramento rispetto ad HF.

possiamo scrivere:

$$|\psi\rangle = C_0 |\psi_{HF}\rangle + \sum_{x,y} C_{x,y} \hat{a}_x \hat{a}_y^\dagger |\psi_{HF}\rangle + \sum_{a,b,c,d} C_{a,b,c,d} \hat{a}_a \hat{a}_b \hat{a}_c^\dagger \hat{a}_d^\dagger |\psi_{HF}\rangle + ... \quad (1.2.2.1)$$

Dove $\hat{a}$ e $\hat{a}^\dagger$ sono gli operatori di eccitazione e diseccitazione fra gli orbitali occupati che HF considerava e gli orbitali liberi.
I coefficienti C si determinano applicando il principio variazionale, soluzione che è formalmente equivalente alla diagonalizzazione di una matrice data una base finita.
Se si considerano tutti i termini di eccitazione possibili (singola, doppia, tripla...) il metodo si dice declinato in *Full Configuration Interaction* (FCI). Le soluzioni FCI forniscono risultati estremamente accurati (fornirebbero i risultati esatti se si usassero basi infinite[7]), ma è impossibile applicare tale metodo per molecole non estremamente semplici poiché il peso computazionale scala come $\mathcal{O}(M^N)$ (M è il numero di orbitali di spin, N il numero di elettroni).
Esistono, dunque, differenti approssimazioni di FCI che limitano il calcolo a un certo numero di eccitazioni (CISD, ad esempio, si limita alle eccitazioni singole e doppie) o permettono di calcolare solo i termini più rilevanti (ad esempio SCF o MCSCF).

1.2.3 Coupled-Cluster theory

Abbiamo visto che può avere senso troncare l'espansione FCI a un certo ordine di eccitazioni per limitare il costo computazione. I metodi di troncamento, però, non sono in grado di conservare la fondamentale proprietà della *size consistecy*: l'energia di una molecola deve essere uguale alla somma dell'energia dei singoli atomi distinti (dunque allontanati). Matematicamente questa proprietà non è scontata, malgrado lo sia fisicamente e, dunque, ogni metodo che la viola è da utilizzare con molta cautela.
Per questo motivo si sono cercate altre strade a partire proprio dalla conservazione di questa proprietà.
La teoria Coupled-Cluster (CC) cerca di mescolare elementi tipicamente propri di una teoria perturbativa come MP, che in effetti preserva la *size consistency*, sia elementi di una soluzione esatta come CI.
L'idea è di scrivere la funzione d'onda $|\psi\rangle$ di stato fondamentale come prodotto fra lo stato risultante da HF e l'operatore di cluster il quale "contiene" le eccitazioni (singole, doppie, triple...) utilizzate anche nel CI.

$$|\psi\rangle = e^{\hat{T}} |\psi_{HF}\rangle \quad (1.2.3.1)$$

$\hat{T}$ è somma degli operatori relativi a diversi ordini di eccitazione: $\hat{T} = \hat{T}_1 + \hat{T}_2 + ...$ Ad esempio:

$$\hat{T}_1 = \sum_{i,m} \theta_i^m a_m^\dagger \hat{a}_i \quad (1.2.3.2)$$

[7] Nonostante ciò la notazione standard parla comunque dei risultati FCI come di risultati esatti.

$$\hat{T}_2 = \frac{1}{2} \sum_{i,j,m,n} \theta_{i,j}^{m,n} a_m^\dagger a_n^\dagger \hat{a}_i \hat{a}_j \tag{1.2.3.3}$$

Ovviamente anche questo metodo viene troncato: le sue varianti più utilizzate sono la CCSD (eccitazioni singole e doppie) e la CCSDT (eccitazioni singole, doppie e triple).
Nel prosieguo di questa tesi utilizzerò, in particolare, la CCSD tramite la quale si sono raggiunti risultati molto simili a quelli ottenuti tramite diagonalizzazione esplicita, in particolare partendo dalla funzione d'onda di HF.
Interessante porre l'attenzione su una particolare identità: nel caso che siano possibili al più doppie eccitazioni, allora CCSD coincide con FCI.
Inoltre, questo tipo di ansatz della funzione d'onda può essere modificato in modo tale che l'operatore di cluster sia unitario. In questo caso si parla di metodi UCC (la U iniziale corrisponde a *Unitary*). Questi metodi sono interessanti perchè possono essere portati direttamente su quantum computer, come vedremo in seguito.

Oltre alla *size consistency* vi è un'altra proprietà che è fondamentale analizzare ed è quella del comportamento variazionale di un metodo. Se un metodo ha tale comportamento allora siamo certi che, qualsiasi energia esso ci restituisca, questa non sia inferiore all'energia reale. Anche tale proprietà non è modellizzabile semplicemente a livello matematico e, in effetti, non tutti i metodi che abbiamo visto (e che pure vengono usati) la rispettano. Idealmente vorremmo un metodo con entrambe le caratteristiche, ma dovendoci accontentare di uno con una sola di queste due, dobbiamo sempre porre attenzione ai problemi che possiamo incontrare (si veda la tabella 1.1).

Metodo	Size Consistency	Comportamento variazionale
Hartree-Fock	✓	✓
MP	✓	✗
FCI	✓	✓
CISD (e CISDT...)	✗	✓
CC (UCCSD...)	✓	✗

Tabella 1.1: Proprietà fondamentali dei metodi classici

Capitolo 2

Soluzioni computazionali quantistiche

Analizzato il problema chimico, cambiamo argomento e passiamo alla computazione quantistica. In questo capitolo inizialmente introdurrò gli elementi e i termini fondamentali per comprendere l'utilizzo di un *quantum computer*, successivamente passerò a un'introduzione del *Variational Quantum Eigensolver*. Completerò il capitolo con un'introduzione a Qiskit, strumento fondamentale per simulare (o utilizzare realmente) un computer quantistico.
Per ulteriori approfondimenti su questo capitolo si veda, in particolare, *Mastering Quantum Computing with IBM QX* [13].

2.1 Computer quantistici

2.1.1 Qubit, gate e circuiti

Qubit e rappresentazioni

Un qubit, l'unità fondamentale di informazione per il *quantum computing*, è un sistema fisico quantistico a due livelli[1] [15].
Dal punto di vista teorico, possiamo rappresentare un qubit come una sovrapposizione di stati:

$$|Q\rangle = \frac{a\,|0\rangle + b\,|1\rangle}{\sqrt{a^2 + b^2}} \tag{2.1.1.1}$$

Essendo un sistema a due stati è evidente che, data una qualsiasi base bidimensionale (si considera per semplicità quella canonica), possiamo descrivere coi due coefficienti $a, b \in \mathbb{C}$ tutti i possibili stati.
Qui sta il fondamento della computazione quantistica: i bit classici (che possono assumere unicamente due valori), vengono sostituiti dai qubit, i quali possono assumere

[1] Interessante notare che è possibile considerare anche sistemi a più livelli sia in linea teorica, sia nei computer quantistici attualmente esistenti[14].

un'infinita combinazione di valori. Graficamente possiamo visualizzare un bit come due singoli punti (lo 0 e l'1) e una freccia che indica l'uno o l'altro; d'altra parte i qubit, invece, possono essere rappresentati tramite una sfera tridimensionale (i tre assi corrispondono ai coefficienti su 3 diverse basi bidimensionali) e una freccia che può indicare uno qualsiasi dei punti della sfera (evidentemente infiniti).
Le tre basi che si usano per rappresentare un qubit su una sfera (si parla di sfera di Bloch, figura 2.1.1) sono: $|0,1\rangle$, $|+,-\rangle$, $|\circlearrowleft,\circlearrowright\rangle$ con:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$
$$|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \qquad |-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$$
$$|\circlearrowright\rangle = \frac{|0\rangle+i|1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\i\end{pmatrix} \qquad |\circlearrowleft\rangle = \frac{|0\rangle-i|1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-i\end{pmatrix}$$

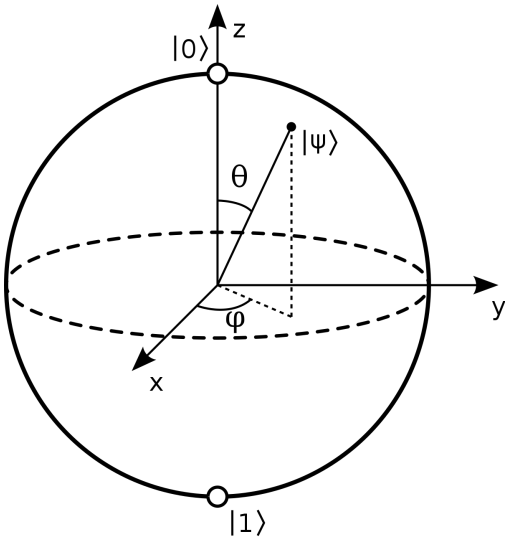


Figura 2.1.1: Sfera di Bloch

Il fatto che un qubit possa assumere un numero infinito di valori, però, non deve illudere troppo: uno dei postulati della meccanica quantistica ci dice subito che, effettuando una qualsiasi misura su un qubit, provochiamo il collasso della funzione d'onda e, sostanzialmente, perdiamo gran parte delle informazioni contenute nel qubit.
In particolare, se consideriamo un qubit sulla base $|0,1\rangle$, potremo ottenere, con una misura, solo lo stato $|0\rangle$ o lo stato $|1\rangle$. La probabilità di ottenere un risultato rispetto all'altro sarà data dal rapporto fra i moduli dei coefficienti a,b dello stato stesso.

Operazioni sui qubit

Le operazioni possibili su un bit classico vengono implementate tramite *classical gates* (in italiano chiamate porte logiche) come ad esempio AND, OR, NOT. Fondamentale è

il concetto di *universal gate set* ovvero un gruppo di porte logiche con cui è possibile, tramite diverse combinazioni, riprodurre tutte le possibili operazioni.
Parallelamente ai *classical gates*, vengono definiti i *quantum gates*, che possono compiere operazioni e trasformazioni sui qubit. Dal momento che un gate deve portare da uno stato valido, esprimibile come vettore di due coefficienti, a un altro, è chiaro che ogni gate può essere espresso in forma matriciale. In particolare, questa sarà una matrice unitaria.
In questa sezione cercherò di illustrare i *gate* più utilizzati.

X gate (NOT)

$$-\boxed{X}-$$

In forma matriciale si indica

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Evidentemente, dunque, l'applicazione sugli stati fondamentali risulta:

$$|0\rangle \; -\boxed{X}- \quad |1\rangle$$

$$|1\rangle \; -\boxed{X}- \quad |0\rangle$$

Il caso più generale, invece:

$$\alpha|0\rangle + \beta|1\rangle \; -\boxed{X}- \quad \alpha|1\rangle + \beta|0\rangle$$

Y gate

$$-\boxed{Y}-$$

In forma matriciale si indica

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Evidentemente, dunque, l'applicazione sugli stati fondamentali risulta:

$$|0\rangle \; -\boxed{Y}- \quad i|1\rangle$$

$$|1\rangle \; -\boxed{Y}- \quad -i|0\rangle$$

Il caso più generale, invece:

$$\alpha|0\rangle + \beta|1\rangle \; -\boxed{Y}- \quad i\alpha|1\rangle - i\beta|0\rangle$$

Z gate

$$-\boxed{Z}-$$

In forma matriciale si indica

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Evidentemente, dunque, l'applicazione sugli stati fondamentali risulta:

$$|0\rangle \text{ —}\boxed{Z}\text{— } |0\rangle$$

$$|1\rangle \text{ —}\boxed{Z}\text{— } -|1\rangle$$

Il caso più generale, invece:

$$\alpha|0\rangle + \beta|1\rangle \text{ —}\boxed{Z}\text{— } \alpha|0\rangle - \beta|1\rangle$$

H gate

In forma matriciale si indica

$$\text{—}\boxed{H}\text{—}$$

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

Evidentemente, dunque, l'applicazione sugli stati fondamentali risulta:

$$|0\rangle \text{ —}\boxed{H}\text{— } \frac{|0\rangle+|1\rangle}{\sqrt{2}}$$

$$|1\rangle \text{ —}\boxed{H}\text{— } \frac{|0\rangle-|1\rangle}{\sqrt{2}}$$

Il caso più generale, invece:

$$\alpha|0\rangle + \beta|1\rangle \text{ —}\boxed{H}\text{— } \frac{\alpha+\beta}{\sqrt{2}}|0\rangle - \frac{\alpha-\beta}{\sqrt{2}}|1\rangle$$

S gate

In forma matriciale si indica

$$\text{—}\boxed{S}\text{—}$$

$$\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$$

Evidentemente, dunque, l'applicazione sugli stati fondamentali risulta:

$$|0\rangle \text{ —}\boxed{S}\text{— } |0\rangle$$

$$|1\rangle \text{ —}\boxed{S}\text{— } i|1\rangle$$

Il caso più generale, invece:

$$\alpha|0\rangle + \beta|1\rangle \text{ —}\boxed{S}\text{— } \alpha|0\rangle + i\beta|1\rangle$$

In particolare possiamo notare che che $S = \sqrt{Z}$.

T gate

In forma matriciale si indica

$$\text{—}\boxed{T}\text{—}$$

$$\begin{pmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{pmatrix}$$

Evidentemente, dunque, l'applicazione sugli stati fondamentali risulta:

$$|0\rangle \text{ —}\boxed{T}\text{— } |0\rangle$$

$$|1\rangle \text{ —}\boxed{T}\text{— } e^{i\frac{\pi}{4}}|1\rangle$$

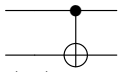
Il caso più generale, invece:

$$\alpha\,|0\rangle + \beta\,|1\rangle \;\text{—}\boxed{T}\text{—}\quad \alpha\,|0\rangle + e^{i\frac{\pi}{4}}\beta\,|1\rangle$$

In particolare possiamo notare che che $T = \sqrt{S} = \sqrt[4]{Z}$.

CNOT gate

Fino ad ora abbiamo visto gate solo su qubit singoli, ma possiamo anche svolgere operazioni fra più qubit. In particolare il CNOT gate (*Controlled NOT*) è fondamentale per un'altra proprietà principale della meccanica quantistica sfruttata dal *quantum computing*: l'entanglement quantistico.
A seconda del valore del primo qubit viene applicato o meno un NOT gate sul secondo qubit.
Viene indicato:



Può essere rappresentato come matrice 4x4:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Risolvendo per i casi principali otteniamo:

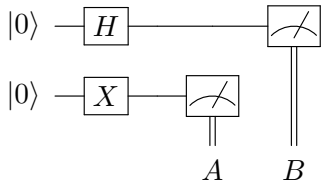
$$\begin{aligned} |00\rangle &\longrightarrow |00\rangle \\ |10\rangle &\longrightarrow |11\rangle \\ |11\rangle &\longrightarrow |10\rangle \\ |01\rangle &\longrightarrow |01\rangle \end{aligned}$$

Set universale

Vi sono diversi set di *quantum gates* che formano un gruppo universale. Sufficientemente semplice è dimostrare che se ne ha uno considerando H, S, T e CNOT.

Quantum circuit

Il corrispettivo di una funzione classica è costituito da una combinazione particolare di gates che si applica a un determinato set di qubit. In questo caso si ha un *quantum circuit* che si può facilmente rappresentare graficamente.
Ogni riga singola rappresenta un qubit che, partendo dallo stato indicato all'estrema sinistra, subisce una serie di trasformazioni mediante i quantum gates. Per effettuare una misura si applica —  che, eventualmente, registra l'informazione su un bit classico.
Ad esempio, un semplice quantum circuit potrebbe essere il seguente:



Dopo un'esecuzione del circuito disegnato avremo il bit A che sarà certamente un 1, mentre il bit B avrà uguale probabilità di essere 0 o 1. Studiando un circuito complesso, per recuperare l'informazione sulle probabilità di misura finali, il singolo circuito sarà eseguito molteplici volte: ad esempio, nel nostro caso, eseguendo varie volte il circuito, ci aspettiamo di avere approssimativamente il medesimo numero di misure (11) e di (10).

Limiti fisici

Per quanto, in linea teorica, possiamo progettare circuiti con centinaia di gate e di qubit, la tecnologia attualmente disponibile è estremamente limitante.
In particolare, i computer di IBM che sono disponibili hanno un basso numero di qubit, da zero a 15, e sono estremamente rumorosi. Vi è anche una limitazione sul numero di gate sempre relativa al rumore: ogni gate introduce una probabilità di errore ed è dunque opportuno limitarne l'uso il più.
Ad esempio, nel caso del circuito sopra descritto, il bit A dovrebbe essere sempre 1, ma per il rumore la situazione cambia; nella realtà ci aspettiamo comunque una piccola probabilità di misurare 0. Il fenomeno che contribuisce maggiormente a questo errore è quello della decoerenza: le informazioni contenute in un qubit (ovvero le fasi relative) hanno una certa possibilità di essere perdute attraverso l'interazione con l'ambiente esterno. Vi sono poi altri errori legati all'implementazione *hardware* dei qubit stessi, dei *gate* e delle modalità di misura e preparazione di uno specifico stato.
Per meglio mostrare la questione appena introdotta, ho eseguito il circuito sopraindicato per 8000 volte su un *quantum computer* reale e su un simulatore perfetto: i risultati riportati in tabella 2.1 mostrano chiaramente la presenza di errori (seppur in numero limitato).

Risultato	Conteggi hardware	Conteggi simulati	Conteggi esatti
00	111	0	0
01	3996	4031	4000
10	113	0	0
11	3780	3969	4000

Tabella 2.1: Confronto fra i risultati ideali per 8000 ripetizioni di un quantum circuit e le misure hardware soggette ad errori

Vi è un'altra limitazione importante: nei computer attuali non tutti i qubit sono fisicamente collegati l'uno con l'altro. Questo è importante nell'applicazione di gate su

più qubit (come il CNOT): si potrà applicare un CNOT *gate* solo fra qubit fisicamente contigui.

2.1.2 Da sistemi fermionici a circuiti di qubit

Dopo la necessaria, ed estremamente limitata, introduzione ai quantum circuit, torniamo al problema originale: serve un modo per scrivere, sotto forma di qubit, uno stato $|\psi\rangle$ (ad esempio ricavato con HF)[16].
La prima cosa che possiamo notare è che lo spazio di Fock[2] per M orbitali di spin ha dimensione $dim(\mathscr{F}) = 2^M$. Ci serve dunque uno spazio di uguale dimensione ottenibile con M qubit: $\mathscr{H}^{\otimes M}$.
È chiaro, a questo punto, che possiamo definire un isomorfismo fra i due spazi:

$$\Phi : \mathscr{F} \to \mathscr{H}^{\otimes M} \tag{2.1.2.1}$$

Tale isomorfismo non è univocamente definito e, anzi, è importante sceglierlo con cura per ottimizzare al meglio il circuito, visti anche i limiti fisici già evidenziati.

Isomorfismo di Jordan-Wigner

Il più intuitivo di questi isomorfismi è quello ideato da Jordan e Wigner.
Come gli altri che introdurrò dopo questo, esso si fonda sul formalismo della seconda quantizzazione[3] che, senza entrare in alcun dettaglio, ci permette di esprimere ogni stato (fermionico o bosonico) tramite gli operatori di creazione ($\hat{a}^\dagger$) e distruzione ($\hat{a}$). Il compito fondamentale degli isomorfismi sarà dunque quello di tradurre in qubit gli operatori $\hat{a}$ e $\hat{a}^\dagger$.
Un sistema a due livelli, come quelli che dobbiamo considerare, ha i due possibili stati identificabili con $|\emptyset\rangle$ e $\hat{a}^\dagger |\emptyset\rangle$. La mappatura più naturale di questi singoli stati è, ovviamente:

$$\begin{aligned} |\emptyset\rangle &\to |0\rangle_q \\ \hat{a}^\dagger |\emptyset\rangle &\to |1\rangle_q \end{aligned}$$

Inoltre, se scriviamo $\left(\hat{a}^\dagger |\psi\rangle\right)_q = \left(\hat{a}^\dagger\right)_q |\psi\rangle_q$, possiamo dire che:

$$\begin{aligned} \hat{a}_q^\dagger |0\rangle &= |1\rangle \\ \hat{a}_q^\dagger |1\rangle &= 0 \end{aligned}$$

[2] Lo spazio di Fock è uno spazio di Hilbert introdotto nel formalismo della seconda quantizzazione. È definito come il prodotto tensoriale degli spazi di Hilbert di singola particella.

[3] In realtà esistono anche alcuni isomorfismi che fanno a meno della seconda quantizzazione, tuttavia questi metodi non sono per nulla efficienti e non verranno trattati in questa tesi.

Evidentemente, dunque, possiamo scrivere l'operatore nella ormai chiara forma matriciale:

$$\hat{a}^\dagger \longleftrightarrow \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = S^+ \tag{2.1.2.2}$$

Possiamo anche ricavare:

$$\hat{a} \longleftrightarrow \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = S^- \tag{2.1.2.3}$$

L'operatore "numero", prodotto dell'operatore di creazione e di quello di distruzione risulta calcolabile:

$$\hat{a}^\dagger \hat{a} \longleftrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{\mathbb{I} - Z}{2} \tag{2.1.2.4}$$

Fino ad ora ci siamo limitati a sistemi con un singolo qubit, ma possiamo rapidamente estendere la nostra rappresentazione considerando un orbitale di spin per ogni qubit ($\otimes$ rappresenta il prodotto di Kronecker):

$$|x_0, x_1, ..., x_{M-1}\rangle \rightarrow |x_0, x_1, ..., x_{M-1}\rangle_q = \bigotimes_{k=0}^{M-1} |x_k\rangle_q \tag{2.1.2.5}$$

Gli operatori di creazione e distruzione, per più qubit, diventano, tralasciando tutti i passaggi intermedi:

$$(a_p)_q = Q_p \otimes Z_{p-1} \otimes ... \otimes Z_0 \tag{2.1.2.6}$$

$$\left(a_p^\dagger\right)_q = Q_p^\dagger \otimes Z_{p-1} \otimes ... \otimes Z_0 \tag{2.1.2.7}$$

Dove gli operatori Q sono così definiti:

$$Q = \frac{1}{2}(X + iY) \quad Q^\dagger = \frac{1}{2}(X - iY) \tag{2.1.2.8}$$

E i pedici agli operatori si riferiscono al qubit sul quale operano.
Gli operatori Q cambiano l'occupazione del singolo orbitale di spin, mentre la stringa dei diversi Z si occupa unicamente di apportare una correzione di fase ± 1 e per questo si dice che 'calcola la parità di uno stato'.
Questo isomorfismo riesce a mappare uno stato di M orbitali di spin in M qubit, mentre invece la soluzione esatta FCI aveva bisogno di $\mathcal{O}(M^N)$ termini: qui si vede chiaramente il motivo per il quale è così promettente l'uso di computer quantistici nel campo della chimica computazionale.
Si noti, tuttavia, che benché l'isomorfismo di Jordan-Wigner sia estremamente semplice e intuitivo e permetta di mappare l'occupazione di uno stato localmente (ovvero è sufficiente misurare un qubit in un singolo punto del quantum circuit per vedere se lo stato è $|\emptyset\rangle$ o $\hat{a}^\dagger |\emptyset\rangle$), gli operatori non sono locali: ad esempio si vede che, nelle equazioni 2.1.2.6 e 2.1.2.7, gli operatori necessitano di gate (gli Z) su vari qubit. Come già detto: più gate abbiamo nel nostro quantum circuit, più è alta la probabilità di errori.

Isomorfismo di Parità

Il secondo isomorfismo utilizzabile è quello di parità.
L'idea è di quella di mappare nei qubit non direttamente l'occupazione di ciascun orbitale, ma la quantità chiamata "parità di un set di numeri di occupazione f_i":

$$p_j = \sum_{i=0}^{j-1} f_i \mod 2 \tag{2.1.2.9}$$

Indichiamo dunque l'isomorismo Φ:

$$|f_{M-1}, ..., f_0\rangle \to |p_{M-1}, ..., p_0\rangle_q \tag{2.1.2.10}$$

L'operazione può essere facilmente espressa in forma matriciale[4]:

$$\begin{pmatrix} p_0 \\ p_1 \\ ... \\ p_{M-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ 1 & 1 & 0 & ... & 0 \\ ... & ... & ... & ... & 0 \\ 1 & 1 & 1 & ... & 1 \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ ... \\ f_{M_1} \end{pmatrix} \tag{2.1.2.11}$$

A questo punto dovremmo ricavare la forma degli operatori di creazione e distruzione che però non sono così immediati come nell'isomorfismo di Jordan-Wigner. Si può tuttavia dimostrare che:

$$\left(\hat{a}_j^\dagger\right)_q |p_{M-1}, ..., p_j, ..., p_0\rangle_q = \delta_{p_j\, p_{j-1}} (-1)^{p_j - 1} |\bar{p}_{M-1}, ..., \bar{p}_j, p_{j-1}, ..., p_0\rangle_q \tag{2.1.2.12}$$

E infine si può scrivere:

$$\left(\hat{a}_j^\dagger\right)_q = \prod_{i=j}^{M-1} X_i \left(\frac{X_j \otimes Z_{j-1} - iY_{j-1}}{2}\right) \tag{2.1.2.13}$$

Questo isomorfismo delocalizza completamente l'informazione sull'occupazione, ma localizza invece la parità. Tuttavia, anche se gli operatori risultano generalmente più localizzati che per Jordan-Wigner, non si ha un reale guadagno di efficienza (per ogni operatore fermionico bisogna comunque aggiungere $\mathcal{O}(M)$ gate).
La trasformazione di parità è interessante perché permette, in taluni casi, di ridurre il numero di qubit (questo tema sarà trattato approfonditamente più avanti).

Isomorfismo di Bravyi-Kitaev

Per diminuire il numero di gate si può utilizzare l'isomorfismo di Bravyi-Kitaev, ancora meno intuitivo della parità, che permette di utilizzare per ogni operatore fermionico solo $\mathcal{O}(\log M)$ gate al prezzo di delocalizzare sia le informazioni sulla parità che sull'occupazione del'orbitale.

[4] Si fa notare che la matrice è facilmente invertibile.

La trasformazione si può indicare per via matriciale:

$$\begin{pmatrix} b_0 \\ ... \\ b_{M-1} \end{pmatrix} = \beta_M \begin{pmatrix} f_0 \\ ... \\ f_{M-1} \end{pmatrix} \tag{2.1.2.14}$$

Dove β_M è una matrice definita ricorsivamente per $M = 2^n$:

$$\beta_{2^{n+1}} = \begin{pmatrix} \beta_{2^n} & 0 \\ A_{2^n} & \beta_{2^n} \end{pmatrix} \ , \ A_{2^n} = \begin{pmatrix} 0 & ... & 0 \\ 0 & ... & 0 \\ ... & ... & ... \\ 0 & ... & 0 \\ 1 & ... & 1 \end{pmatrix} \tag{2.1.2.15}$$

La mappatura degli operatori fermionici su qubit è ancora meno intuitiva, ma ci permette di usare decisamente meno gate, come detto sopra.

$$\left(\hat{a}_j^\dagger\right)_q = \frac{1}{2} X_{U_M(j)} \left(X_j Z_{P_M(j)} - iY_j Z_{S_M(j)}\right) \tag{2.1.2.16}$$

Dove abbiamo utilizzato tre set predefiniti di qubit:

- $P_M(j)$: detto "Parity Set", contiene i qubit che hanno registrata l'informazione di parità del set $f_{j-1}...f_0$, sono in numero minore;
- $U_M(j)$: detto "Update Set", contiene i qubit (ad esclusione di j) che devono essere invertiti quando si applica $\hat{a}_j^\dagger$;
- $F_M(j)$: detto "Flip Set", contiene i qubit che è necessario conoscer per determinare se b_j è uguale o meno a f_j.

A titolo d'esempio vengono riportate in tabella 2.2 le mappature di un generico stato a un singolo elettrone su quattro orbitali, nei tre isomorfismi.

Fermioni	Jordan-Wigner	Parità	Bravyi-Kitaev
$a\|0001\rangle + b\|0010\rangle +$ $+c\|0100\rangle + d\|1000\rangle$	$a\|0001\rangle + b\|0010\rangle +$ $+c\|0100\rangle + d\|1000\rangle$	$a\|1111\rangle + b\|1110\rangle +$ $+c\|1100\rangle + d\|1000\rangle$	$a\|1011\rangle + b\|1010\rangle +$ $+c\|1100\rangle + d\|1000\rangle$
$\hat{a}_0$	Q_0	$X_3X_2X_1Q_0$	$X_3X_1Q_0$
$\hat{a}_1$	Q_1Z_0	$X_3X_2\left(Q_1\|0\rangle\langle 0\|_0 - Q_1^\dagger\|1\rangle\langle 1\|_0\right)$	$X_3\left(Q_1\|0\rangle\langle 0\|_0 - Q_1^\dagger\|1\rangle\langle 1\|_0\right)$
$\hat{a}_2$	$Q_2Z_1Z_0$	$X_3\left(Q_2\|0\rangle\langle 0\|_1 - Q_2^\dagger\|1\rangle\langle 1\|_1\right)$	$X_3Q_2Z_1$
$\hat{a}_3$	$Q_3Z_2Z_1Z_0$	$Q_3\|0\rangle\langle 0\|_2 - Q_3^\dagger\|1\rangle\langle 1\|_2$	$\frac{1}{2}\left(Q_3\left(\bar{\bar{1}} + Z_2Z_1\right) - Q_3^\dagger\left(\bar{\bar{1}} - Z_2Z_1\right)\right)$

Tabella 2.2: Differenze fra gli isomorfismi di Jordan-Wigner, di parità e di Bravyi-Kitaev

Hamiltoniana Fermionica

Fino ad ora abbiamo solo parlato del mappare stati ed operatori di creazione e distruzione su qubit; abbiamo detto che il formalismo della seconda quantizzazione ci permette

di esprimere ogni stato con combinazioni particolari di questi operatori, ma ciò è limitante: infatti anche l'Hamiltoniana del sistema si può scrivere tramite questi operatori. Potendoli "tradurli" in qubit, quindi, possiamo anche mappare l'intera Hamiltoniana di un sistema su un quantum circuit; in particolare esso risulta dato da una combinazione lineare di prodotti fra operatori di Pauli (X,Y,Z,I) di singolo qubit:

$$\hat{H} = \sum_j h_j \bigotimes_{i=0}^{M-1} \sigma_i^j \tag{2.1.2.17}$$

Dove h_j è un coefficiente reale e σ_i^j è uno degli operatori di Pauli.
Per linearità, dato uno stato $|\psi\rangle$:

$$\langle\psi| \hat{H} |\psi\rangle = \sum_j \langle\psi| \left(\bigotimes_{i=0}^{M-1} \sigma_i^j \right) |\psi\rangle \tag{2.1.2.18}$$

Il problema relativo al calcolo del valore di aspettazione di un'Hamiltoniana, dunque, è ridotto al calcolo del valore di aspettazione di operatori di Pauli su singoli qubit.
Misurando un qubit in sovrapposizione di $|0\rangle$ e $|1\rangle$ si ottiene ovviamente solo $|0\rangle$ o $|1\rangle$ che sono, in realtà, gli autovalori di Z. Per misurare autovalori degli altri operatori devo applicare la giusta rotazione al vettore nella sfera di Bloch. Si veda la tabella 2.3 per la traduzione in *quantum gate* degli operatori di Pauli.

Pauli	Qubit Gate
Z	I
X	H
Y	HS^+

Tabella 2.3: Mappatura in quantum gate degli operatori di Pauli

Si può dimostrare che un'Hamiltoniana così implementata, per un operatore di Pauli a n qubit, rimane molto efficiente scalando (in profondità) come $\mathcal{O}(n)$; mentre il numero di operatori di Pauli va come $\mathcal{O}(M^4)$ per un sistema a M qubit.

2.1.3 Riduzione del numero di qubit

Nell'era NISQ dobbiamo affrontare il problema pratico della ridotta dimensione dei computer disponibili (in genere possiamo dire che c'è un limite di 5 qubit).
Abbiamo, quindi, la necessità di limitare il numero di qubit necessari alla risoluzione dei nostri problemi: da questo nasce il concetto di *tapering* (assottigliamento/riduzione) di qubit che coinvolge differenti tecniche [17].

Riduzione di due qubit con trasformazione di parità o di Bravyi-Kitaev

Il metodo più semplice per ridurre il numero di qubit necessari viene "gratuito" con l'utilizzo dell'isomorfismo di parità (e di Bravyi-Kitaev) precedentemente spiegato (si veda la sezione 2.1.2).

Si sfrutta il fatto che un'Hamiltoniana fermionica conserva (in caso di un sistema non relativistico e non patologico) il numero di particelle con spin fissato. Se chiamiamo p_α il set di orbitali di spin up e p_β il set di orbitali di spin down:

$$\hat{N}_\alpha = \sum_{p_\alpha} \hat{a}^\dagger_{p_\alpha} \hat{a}_{p_\alpha} \ , \ \hat{N}_\beta = \sum_{p_\beta} \hat{a}^\dagger_{p_\beta} \hat{a}_{p_\beta} \tag{2.1.3.1}$$

I due operatori $\hat{N}_\alpha$ e $\hat{N}_\beta$ sono costanti del moto. Il fatto che commutino con l'Hamiltoniana, dunque, può essere sfruttato per eliminare due qubit. Se, infatti, si scrive esplicitamente un circuito si può immediatamente notare (o si può dimostrare) che due qubit sono fissati a un valore invariante e quindi possono essere trascurati. Questo meccanismo viene in genere indicato come *two-qubits reduction*

Frozen core

Un altro metodo intuitivo può essere quello di ridurre il numero di orbitali considerati. Evidentemente, infatti, se interessa solo il comportamento chimico di una certa molecola, gli orbitali più interni sono inerti o influenti in misura minore di quelli di valenza. Da questa considerazione nasce l'idea del *frozen core*, ovvero l'idea di considerare gli orbitali interni congelati e "statici" risparmiando i qubit necessari a descriverli[5].

Simmetrie

Oltre alle costanti del moto descritte nella "riduzione di due qubit" un sistema può avere varie altre simmetrie come riflessioni discrete e parità.
Anche in questi casi ci si può ricondurre a un autospazio di operatori conservati per diminuire il numero di qubit di qualche unità.

2.1.4 Tecniche di *error correction/mitigation*

Come già detto in più occasioni, uno dei limiti dei *quantum computer* contemporanei è legato agli errori che necessariamente vengono introdotti.
Vi sono alcune soluzioni[18], parziali, pensate per implementazioni immediate o future e sia *hardware* che *software*.

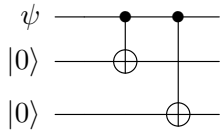
Soluzioni hardware

Le soluzioni *hardware* [20, 19] sono estremamente promettenti, ma attualmente poco applicabili. Sfruttano un alto numero di qubit e sono dunque problematiche per i NISQ; tuttavia saranno probabilmente la norma con computer più grandi.
La cosa più semplice che possiamo fare è triplicare il numero di qubit presenti nel circuito:

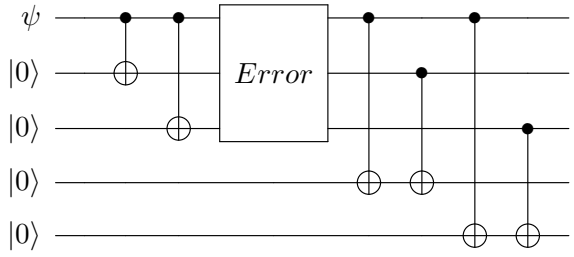
$$|0\rangle \to |000\rangle \ , \ |1\rangle \to |111\rangle$$

[5] Questa tecnica di riduzione non verrà utilizzata in questa tesi poiché mi concentrerò sulla molecola di H_3^+ che non ha orbitali interni.

Una tale operazione è facilmente riproducibile su un quantum circuit[6]:



A questo punto, se un errore che coinvolge un singolo qubit porta ψ in $X\psi$, siamo in grado di recuperare il valore contenuto nel qubit originale osservando il valore di maggioranza. Per identificare un qubit invertito servono altri due qubit ausiliari:



I due qubit finali contengono l'informazione sull'errore:

Qubit 1,2,3	Qubit 4,5
$\|000\rangle$, $\|111\rangle$	$\|00\rangle$
$\|010\rangle$, $\|101\rangle$	$\|10\rangle$
$\|001\rangle$, $\|110\rangle$	$\|01\rangle$
$\|100\rangle$, $\|011\rangle$	$\|11\rangle$

A questo punto possiamo, nei quattro casi riportati in tabella, applicare nuovamente un gate X al qubit "sbagliato".

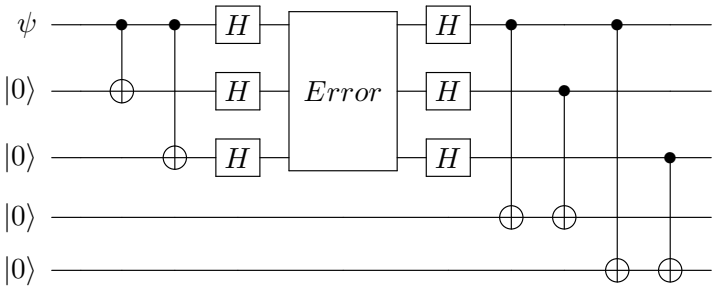
- Se misuriamo $|00\rangle$ non facciamo nulla;
- Se misuriamo $|10\rangle$ applichiamo X al secondo qubit;
- Se misuriamo $|01\rangle$ applichiamo X al terzo qubit;
- Se misuriamo $|11\rangle$ applichiamo X al primo qubit.

Questo per correggere un errore "X" (*flipped* qubit). Se, invece, volessimo correggere un errore "Z" (*phase inverted* qubit) potremmo procedere in maniera simile considerando:

$$|0\rangle \rightarrow |+++\rangle \ , \ |1\rangle \rightarrow |---\rangle$$

e un circuito complessivo:

[6]Onde evitare confusioni, non stiamo violando il teorema di no-cloning in quanto i qubit risultano *entangled*.



In ugual modo procediamo alla correzione, applicando Z invece che X al qubit "sbagliato".
Un computer quantistico con correzione degli errori *hardware* ha, al posto dei singoli qubit che abbiamo visto fino ad ora, dei qubit logici composti da più di un singolo qubit. In particolare possiamo definire due qubit logici prendendo gli ultimi due circuiti disegnati: uno per la correzione X e uno per la correzione Z.

Soluzioni software

Quanto visto per le soluzioni hardware rientra nelle tecniche di *error correction* in quanto i circuiti tendono, appunto, a correggere in automatico gli errori che si vengono a presentare. D'altra parte ora illusterò alcune tecniche di *error mitigation*[21]: metodi che, dati degli errori sistematici, forniscono degli strumenti per attenuarne l'importanza.
Gli errori possono essere di tipi differenti, in particolare possiamo identificare gli errori SPAM (*State Preparation And Mesurement*) che non tengono in considerazione, ad esempio, errori relativi ai *gate*, ma solo errori relativi ai sistemi di misura e prearazione iniziale di stati.
Gli errori SPAM possono essere mitigati con SPAMEM (SPAM *Errors Mitigation*): tale metodo necessita di una calibrazione con l'esecuzione di 2^M (M numero di qubit) circuiti. L'assunto è che una differenza δf nella frequenza dei risultati di una certa misura, rispetto al risultato teorico, si ripresenti immutata in ogni circuito che fa uso dei medesimi qubit. Dall'esecuzione di circuiti di calibrazione otteniamo una matrice di risultati che può essere utilizzata per ricavare le frequenze vere di determinati risultati da quelle misurate. Questo metodo è molto interessante e, in Qiskit, facilmente applicabile, ma parte dall'approssimare tutti gli errori ad errori di tipo SPAM.
In un caso più generale si può utilizzare una tecnica GEM (*General Errors Mitigation*). Tale tecnica è quella che possiamo utilizzare in un caso generale dove non sappiamo a priori dove e quali siano le cause del rumore incontrato e dove, dunque, non siamo in grado di creare un modello matematico di rumore (che sarebbe "invertibile"). Dunque, per illustrare brevemente il metodo, consideriamo la calibrazione di un circuito C_i con una profondità D su M qubit. Come per SPAMEM, prendiamo 2^M circuiti di calibrazione e li riempiamo con tutti i gate del circuito C_i fino a $D/2$ e con gli inversi di quei gate (in ordine opposto). Misurando i risultati di tutti i circuiti si ottiene una prima matrice di calibrazione che andrà poi mediata con una matrice ottenuta in ugual modo, ma considerando i gate della seconda metà del circuito.

Sia per SPAMEM che per GEM, una volta ottenuta una matrice di calibrazione K la si applica con (v sono le frequenze misurate e X è un vettore inizializzato casualmente a norma unitaria) :

$$f(x) = \sum_{i=1}^{2^M} (v_i - K \cdot X)^2 \tag{2.1.4.1}$$

Con la formula 2.1.4.1 possiamo quindi ricavare le frequenze f dalla matrice di calibrazione e dalle frequenze misurate. Con l'approssimazione che gli errori si presentano in modo costante e uniforme, le frequenze f corrispondono ai risultati privi d'errore.

2.2 Variational Quantum Eigensolver[22]

Il *Variational Quantum Eigensolver*[8, 24, 23] è un algoritmo ibrido che sfrutta sia le possibilità offerte dai processori classici (per problemi già risolvibili in tempo polinomiale[7] dall'informatica classica) sia le potenzialità dei processori quantistici.
L'idea di fondo è quella di sfruttare il principio variazionale per minimizzare l'energia di una molecola cambiandone la sua configurazione elettronica.
L'algoritmo si compone di più fasi, data una particolare molecola:

1. Tenendo fissi gli ioni, viene calcolata l'Hamiltoniana corrispondente alla molecola;
2. Si considera un ansatz variazionale della funzione d'onda che descrive il sistema. Questo ansatz sarà espresso come *quantum circuit* composto da rotazioni parametrizzate;
3. Vengono misurati il valore di aspettazione di H e le sue derivate rispetto ai parametri variazionali attraverso il quantum computer;
4. Un ottimizzatore classico sfrutta queste informazioni per variare i valori dei parametri dell'ansatz in modo da minimizzare l'energia;
5. i punti 3 e 4 vengono reiterati fino a raggiungere un minimo dell'energia.

Il *Variational Quantum Eigensolver* ha l'obiettivo di approssimare autovalori e autovettori di una data Hamiltoniana[25]. L'approccio variazionale ci consente di dire che, per qualsiasi funzione $|\psi\rangle$, l'energia calcolata con 2.2.0.1 sia maggiore o uguale all'energia di stato fondamentale.

$$E = \frac{\langle\psi|\,\hat{H}\,|\psi\rangle}{\langle\psi|\,\psi\rangle} \longrightarrow E \geq E_{GS} \tag{2.2.0.1}$$

In VQE, inoltre, lo stato $|\psi\rangle$ è ottenuta a partire da una stato iniziale $|\psi_0\rangle$ a cui viene applicato un operatore variazionale:

$$|\psi\rangle = \hat{U}(\theta)\,|\psi_0\rangle \tag{2.2.0.2}$$

[7] Per un algoritmo eseguibile in tempo polinomiale si intende un algoritmo che completa la sua esecuzione in un tempo che dipende in modo polinomiale dalla dimensione dell'input ad esso necessario.

2.2.1 Funzione costo

Nel VQE è fondamentale definire un'adeguata funzione costo da minimizzare. Usualmente si utilizza l'energia (e dunque l'operatore hamiltoniano) come descritto in equazione 2.2.0.1, ma nulla ci vieta di utilizzare altri operatori con proprietà specifiche; ad esempio è possibile definire operatori che "costringano" la conservazione del numero di particelle del sistema anche se l'ansatz utilizzato (argomento poi trattato al paragrafo 2.2.2), non ha un chiaro fondamento chimico.
Indipendentemente dal "significato" che la funzione costo possiede, essa deve rispettare alcune proprietà per essere valida:

- si deve avere certezza che sia minima nella soluzione corretta;
- si deve essere in grado di calcolarne in modo efficiente il valore di aspettazione su un quantum computer;
- si deve essere in grado di migliorare in modo efficiente il costo (ad esempio variando i parametri della forma variazionale).

Inoltre, non necessariamente, può essere utile che la funzione costo sia *operationally meaningful*, ovvero che il costo computazionale diminuisca con l'avvicinarsi alla soluzione corretta.
Altro importante aspetto da tenere in considerazione è la possibilità d'implementazione nel breve termine: bisogna sempre ricordare che i NISQ, infatti, sono estremamente limitati (si veda la sezione 2.1.4).

2.2.2 Ansatz

Per ansatz si intende quella forma variazionale $\hat{U}(\theta)$ introdotta all'equazione 2.2.0.2 che "si occupa" di portare lo stato iniziale in un generico altro stato. Esiste una grandissima varietà di ansatz che possono essere divisi in due grandi categorie: gli ansatz *problem inspired* che partono da una descrizione teorica del caso specifico analizzato e gli ansatz *problem agnostic* che mirano unicamente a variare lo stato dato, indipendentemente dal problema in questione.
Al primo gruppo appartiene UCCSD (già discussa in sezione 1.2.3) forma variazionale evidentemente basata sulla chimica che in *quantum computing* prende il nome di qUCCSD.
Al secondo gruppo, invece, appartengono altri due utili ansatz che utilizzerò in questa tesi (TwoLocal e SO(4)). Entrambi ricadono anche sotto la categoria degli ansatz *hardware efficient*: essi nascono, infatti, con il preciso scopo di diminuire il numero di gate (e dunque la possibilità d'errore) per rispondere alle limitazioni dei NISQ.

qUCCSD[26, 27]

Abbiamo detto in alla sezione 1.2.3 che l'ansatz per il modello UCCSD è:

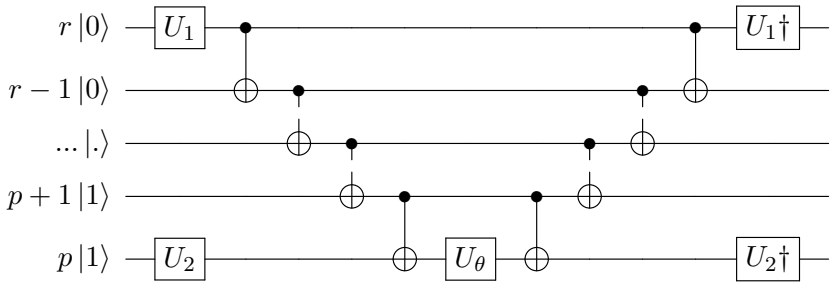
$$|\psi\rangle = e^{\hat{T}-\hat{T}^\dagger} |\psi_0\rangle \tag{2.2.2.1}$$

Dove $\hat{T}$ e $\hat{T}^\dagger$ sono somma degli operatori di singola e doppia eccitazione (e diseccitazione):

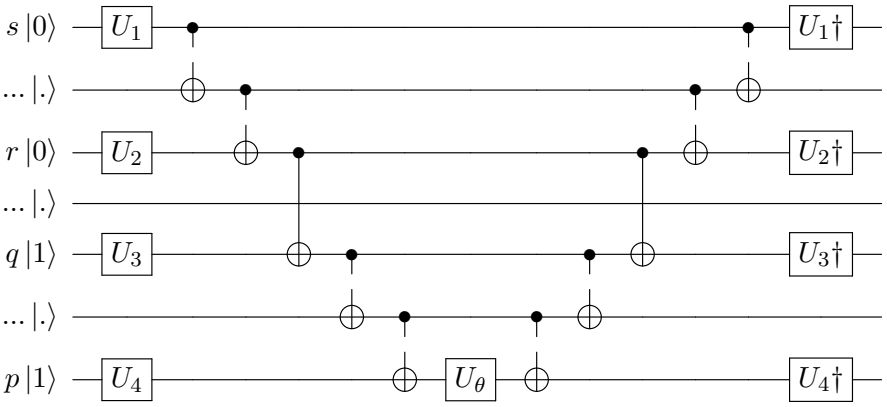
$$\hat{T}_1 = \sum_{i,m} \theta_i^m a_m^\dagger \hat{a}_i \tag{2.2.2.2}$$

$$\hat{T}_2 = \frac{1}{2} \sum_{i,j,m,n} \theta_{i,j}^{m,n} a_m^\dagger a_n^\dagger \hat{a}_i \hat{a}_j \tag{2.2.2.3}$$

A rendere complessa la mappatura di questa variazionale su un computer quantistico è l'esponenziale, i circuiti risultano
per la singola eccitazione, con $(U_1, U_2) = \{(Y, H), (H, Y)\}$:



per la doppia eccitazione con, $(U_1, U_2, U_3, U_4) = \{(H, H, Y, H), (Y, H, Y, Y), (H, H, H, Y), (Y, H, H, H), (H, Y, H, H), (Y, Y, Y, H), (Y, Y, H, Y)\}$:



Gli indici p,q si riferiscono a orbitali virtuali, r ed s a orbitali occupati. In linee tratteggiate sono indicati elementi ripetuti attraverso vari qubit. Di seguito la spiegazione e definizione degli U-gate e dei CNOT-gate.
La forma variazionale ha come elemento fondante $U(\theta)$, che ha la forma:

$$\hat{U}(\boldsymbol{\theta}) = e^{\hat{T}(\boldsymbol{\theta}) - \hat{T}^\dagger(\boldsymbol{\theta})} =$$

$$= \exp\left(\sum_{ij} \theta_{ij} \hat{a}_i^\dagger \hat{a}_j + \sum_{ijkl} \theta_{ijkl} \hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_l - \sum_{ij} \theta_{ij} \hat{a}_j^\dagger \hat{a}_i - \sum_{ijkl} \theta_{ijkl} \hat{a}_l^\dagger \hat{a}_k^\dagger \hat{a}_j \hat{a}_i\right) \quad (2.2.2.4)$$

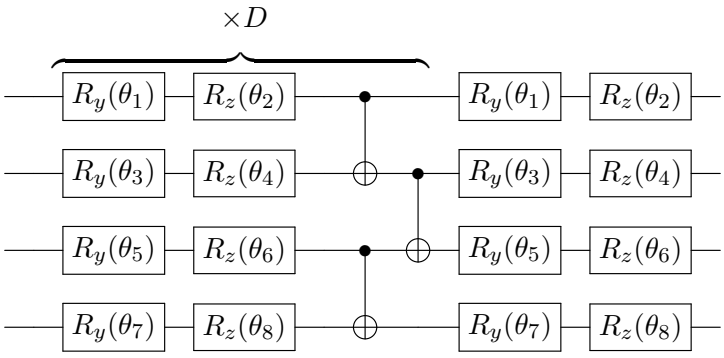
Dove $\boldsymbol{\theta} = \{\{\theta_{ij}\}, \{\theta_{ijkl}\}\}$.
A questo punto possiamo applicare l'approssimazione di Trotter ovvero: $e^{\hat{A}+\hat{B}} \approx e^{\hat{A}} e^{\hat{B}}$.

$$\hat{U}(\boldsymbol{\theta}) = \prod_{ij} \exp\left(\theta_{ij}(\hat{a}_i^\dagger \hat{a}_j - \hat{a}_j^\dagger \hat{a}_i)\right) \times \prod_{ijkl} \exp\left(\theta_{ijkl}(\hat{a}_i^\dagger \hat{a}_j^\dagger \hat{a}_k \hat{a}_l - \hat{a}_l^\dagger \hat{a}_k^\dagger \hat{a}_j \hat{a}_i)\right) \quad (2.2.2.5)$$

Solo a questo punto possiamo mappare sui qubit la forma variazionale con le trasformazioni descritte alla sezione 2.1.2.

TwoLocal

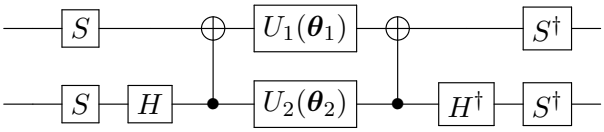
Il circuito di una qUCCSD è estremamente profondo e prevede l'uso di un numero di CNOT eccessivi per i rate d'errori che troviamo nei computer quantistici attualmente esistenti. Sorge la necessità di individuare un circuito che sia in grado di variare efficientemente (con pochi gate) un determinato stato: un ansatz *hardware efficient*[28]. Ad esempio un semplice circuito per un *hardware efficient* ansatz può essere composto da un'alternanza di gate rotazionali e gate di entaglement (si utilizzerà un insieme di CNOT in configurazioni differenti a seconda della connettività hardware disponibile). Stiamo sostanzialmente considerando una rotazione nello spazio SU(2) di un singolo qubit che poi "espandiamo" coi CNOT a tutto lo spazio. I gate rotazionali sono composti da $(\bigotimes R_i(\theta))$ dove R_i è R_y o R_z (si può partire con un singolo gate R_y o metterne una combinazione particolare).
L'insieme di gate di rotazione e di entaglement possono essere ripetuti D volte (D è la profondità della forma variazionale) per una migliore convergenza.
In tutta la tesi chiamerò questo ansatz "TwoLocal" dal nome della classe in Qiskit che lo descrive, alternativamente avrei potuto più semplicemente usare "ansatz rotazionale".
In ogni caso, un ansatz di questo tipo potrebbe essere:



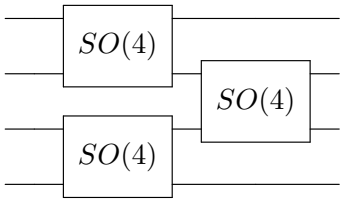
dove D indica il numero di volte che l'insieme di gates può essere ripetuto, componendo la depth del circuito.

SO(4)

Un altro ansatz "efficiente" è SO(4). Esso si basa sul fatto che tutti i gate che conosciamo coinvolgono, al più, due qubit; da questo presupposto si può definire la rotazione più generale nello spazio di Hilbert a quattro dimensioni considerando che i gate di rotazione sono, in realtà, di singolo qubit. Imponendo, inoltre, la condizione per cui i coefficienti finali devono essere reali si può ridurre il numero di parametri per una singola forma variazionale e, infine, si ha:



Questo è il gate SO(4), un ansatz con connettività lineare, per un sistema a quattro qubit, sarebbe:



Gli ottimizzatori classici

Come già detto più volte, VQE necessita di un *optimizer* classico per trovare i migliori parametri per la relativa forma variazionale.
Esistono moltissimi tipi di ottimizzatori, ereditati anche dalle tecniche di *machine learning*. In questa tesi esporrò solo quelli che ho utilizzato nello studio del mio problema:

- **COBYLA:** il *Constrained Optimization BY Linear Approximations* è un algoritmo di ottimizzazione classica (vincolato) utilizzato per funzioni a gradiente sconosciuto.

  É importante notare che COBYLA calcola il valore della funzione stessa un'unica volta per iterazione (indipendente dalla complessità dell'ottimizzazione) ed è dunque molto efficiente in situazioni senza rumore. Se, invece, abbiamo un'ottimizzazione "rumorosa" COBYLA non è consigliato.

- **CG:** il *Conjugate Gradient* è un algoritmo di ottimizzazione iteratativo da utilizzare per sistemi di equazioni lineari a matrice simmetrica e definita positiva. Sostanzialmente CG cerca ad ogni iterazione un set di vettori mutuamente ortogonali (con prodotto definito dalla matrice). A questo punto il set di vettori si può

considerare una base dello spazio e si può riscrivere il problema in termini di tale base, trovando degli autovalori sempre più vicini alla soluzione desiderata.

- **SPSA:** il *Simultaneous Perturbation Stochastic Approximation* è un metodo che, al contrario dei precedenti, non è *Gradient Descent*[8] (GD); SPSA è un'approssimazione di un GD, in quanto calcola solo una derivata parziale casuale fra quelle che compongono il gradiente, risparmiando prezioso tempo computazionale. Per quanto si evitino molti calcoli, il problema non sempre è adatto a un approccio stocastico di questo tipo che comunque è utilizzatissimo in specifici set di problemi.

2.3 Qiskit e IBMQ

Per effettuare tutte le simulazioni di circuiti quantistici si utilizzerà Qiskit: un framework di programmazione per Python totalmente open source. Qiskit non è l'unica possibilità, ma essendo gestito da IBM ha il pregio di rendere semplice l'interazione con i simulatori e i quantum computer reali che l'azienda stessa offre.

2.3.1 I computer quantistici IBM

IBM mette a dispozione diversi quantum computer, tuttavia la maggior parte non è "aperta al pubblico", ma utilizzata solo per la ricerca. Il mio account (di base) ha a disposizione i nove diversi quantum device visibili in tabella 2.4[9].

Nome	Numero di qubit	Quantum volume	Tipo di processore
ibmq_manila	5	32	Falcon r5.11L
ibmq_santiago	5	32	Falcon r4
ibmq_athens	5	32	Falcon r4
ibmq_belem	5	16	Falcon r4
ibmq_quito	5	16	Falcon r4
ibmq_16_melbourne	15	8	Canary r1.1
ibmq_lima	5	8	Falcon r4
ibmq_5_yorktown	5	8	Canary r1
ibmq_armonk	1	1	Canary r1.2

Tabella 2.4: I quantum computer a uso libero e le loro caratteristiche

Il *Quantum Volume* è una misura olistica delle performance di un quantum computer[29]. Ad esempio, possiamo notare che i computer nella parte alta della tabella hanno un alto QV e sono quindi, in linea di massima, da preferire.

[8] Un metodo *Gradient Descent* cerca il minimo di una funzione calcolando il gradiente in un determinato punto e "seguendo" la direzione negativa del gradiente stesso.

[9] Alla consegna di questa tesi è stata annunciata la prossima chiusura di ibmq_16_melbourne e ibmq_athens.

In questa tesi il lavoro è stato condotto principalmente su "ibmq_santiago". Sul sito www.quantum-computing.ibm.com è visibile per ogni device una scheda dettagliata.
La prima cosa che si può notare è una mappa (riportata in figura 2.3.1) che rappresenta i collegamenti presenti fra diversi qubit; ciò è particolarmente importante per l'applicazione di gate a più qubit come CNOT. Si vede che ibmq_santiago è un device lineare, ma ciò non è la norma: esistono computer ciclici o con strutture più articolate ancora. Ad esempio ibmq_montreal (27 qubit, 128 QV), uno dei computer riservati, ha una mappa più complessa visibile, insieme a quella di ibmq_paris, ibmq_toronto e ibmq_johannesburg, in figura 2.3.2.

Figura 2.3.1: Mappa del quantum computer ibmq_santiago

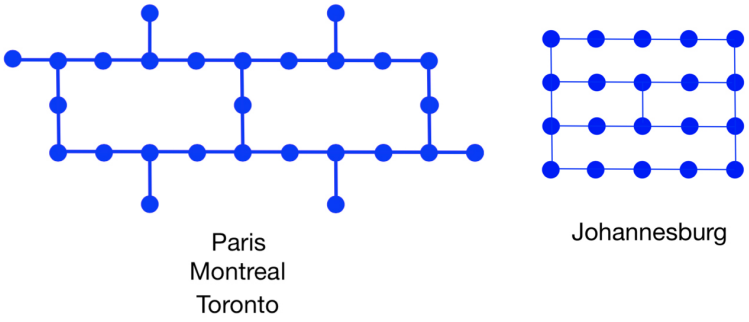


Figura 2.3.2: Mappe 'complesse' di quantum computer

Interessante, inoltre, notare l'indicazione:

Basis gates: CX, ID, RZ, SX, X

Questo è il set universale di gate implementati fisicamente del device; informazione che può essere particolarmente utile per studi di calibrazione, ad esempio, ma che per questa tesi è totalmente indifferente.
Inoltre leggiamo:

Avg. CNOT Error: 6.604e-3
Avg. Readout Error: 2.164e-2

Dunque abbiamo indicazioni per le due maggiori fonti di rumore: i CNOT (gli altri gate sono molto più silenziosi) e le misure stesse.
Infine leggiamo:

Avg. T1: 144.87 us
Avg. T2: 135.81 us

Entrambi i valori sono relativi al fenomeno della decoerenza. T1 è il *relaxation time* che è la costante temporale poissoniana relativa a un qubit che passa spontaneamente da $|1\rangle$ a $|0\rangle$. T2 è il *dephasing time* relativo a cambiamenti spontanei di fase come quello di un qubit che da $|+\rangle$ passa a $|-\rangle$.
Per quanto non strettamente relativo all'hardware, è interessante anche notare l'indicazione:

Total pending jobs: 121

Questo numero (chiaramente variabile) è molto interessante perché mostra come funziona l'utilizzo dei quantum computer. Chiunque voglia, infatti, può inviare a un determinato device il proprio circuito che verrà messo in coda ed eseguito appena possibile. I quantum computer non sono velocissimi nell'esecuzione dei programmi anche perché vanno tendenzialmente eseguiti migliaia di volte, ma il tempo principale che viene usato per effettuare una vera e propria *quantum run* sarà il tempo in coda.

2.3.2 La struttura di Qiskit

Qiskit è diviso principalmente in 5 moduli:

- **Qiskit Terra:** contiene tutte le funzioni necessarie per comporre programmi a partire dai circuiti;
- **Qiskit Aer:** contiene simulatori e modelli di rumore;
- **Qiskit Ignis:** contiene le funzioni riguardanti il rumore e le possibili correzioni;
- **Qiskit Aqua:** contiene una collezione notevole di algoritmi per computer quantistici;
- **Qiskit IBMQ:** contiene le funzioni per accedere all'hardware.

Nei mesi in cui sto scrivendo questa tesi, Qiskit ha parzialmente riorganizzato la sua struttura in moduli introducendo Qiskit Nature che raccoglie tutte le funzioni utili allo studio di chimica quantistica e scienze naturali in generale. L'ambito che questa tesi copre, dunque, è stato specificatamente interessato da questa riorganizzazione. Ho utilizzato, però, una versione di Qiskit più "vecchia" (versione 0.24.1 di Marzo 2021) per evitare problemi di incompatibilità o bug ancora irrisolti.

2.3.3 Backends and Noise Model

Qiskit fornisce vari *backend* per eseguire i circuiti. Sostanzialmente un *backend* è un simulatore particolare o un *quantum computer* specifico.
Ho già introdotto brevemente le differenze fra i vari computer nella sezione 2.3.1, meno intuitiva è la presenza di differenze fra i simulatori.

QASM simulator

Il nome sta per *Quantum Assembly Simulator* (in citazione al noto linguaggio di programmazione tradizionale) ed è il principale backend fornito da Qiskit. Esso emula l'esecuzione dei circuiti eseguendoli molteplici volte (dette *shot*) per ricavare le informazioni sulla sovrapposizione di stati che viene persa nella singola misura. QASM simulator è molto interessante, inoltre, perché estremamente configurabile: si può, ad esempio, andare a simulare anche un modello di rumore (ricavato da un computer reale o creato da zero) che consideri il rate di errori di ogni singolo gate, così come gli errori SPAM o la stessa configurazione fisica di un *quantum computer* (la connettività fra gate).

Statevector simulator

Lo *statevector simulator*, al contrario di QASM, non ottiene i risultati finali (intesi come le probabilità di misura) ripetendo l'esecuzione del circuito, ma invece studia man mano le ampiezze di probabilità considerando un computer privo di errori. È quindi considerabile una simulazione esatta del sistema, intesa come matrici unitarie (i gate) applicate a vettori (gli stati dei qubit). La dimensione delle matrici utilizzate da queste simulazioni scala esponenzialmente con il numero dei qubit.

Capitolo 3

Simulazioni e misure

3.1 Studio iniziale di H_3^+ [30, 31]

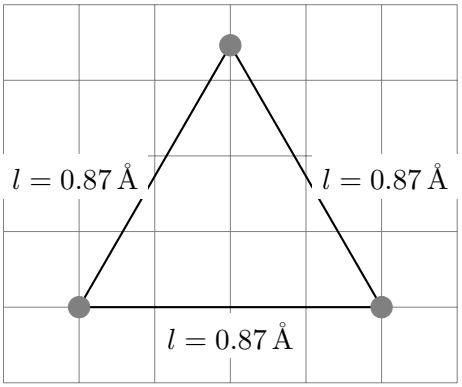
La parte centrale della tesi riguarda lo studio dell'energia molecolare di dissociazione per la molecola di H_3^+. Tale molecola è chiamata "catione idrogenonio" ed è uno ione estremamente abbondante nell'universo, in particolare nello spazio interstellare dove la bassa densità di materia riduce la probabilità di dissociazione della molecola[32].
La reazione che porta alla creazione dell'idrogenonio è:

$$H_2^+ + H_2 \rightarrow H_3^+ + H \tag{3.1.0.1}$$

Mentre l'usuale forma di dissociazione è:

$$H_3^+ \rightarrow H_2 + H^+ \tag{3.1.0.2}$$

La forma dell'idrogenonio è ormai nota[33]: i tre atomi di idrogeno si distribuiscono ai vertici di un triangolo equilatero di lato $l \approx 0.87\,\text{Å}$.



Nella sezione 3.2.1 saranno analizzate altre possibili configurazioni ioniche: in particolare è possibile che la configurazione di stato fondamentale sia lineare come, ad esempio, la simile molecola neutra di H_3[1].

[1] In effetti per un certo tempo si pensò che anche H_3^+ avesse una geometria lineare, come si può vedere in [34].

L'energia minima calcolata analiticamente con la migliore precisione riporta[35]:

$$E_{min} = -1.34383562502\, E_H$$

L'energia di dissociazione (valore sperimentale), invece, risulta[36]:

$$E_{\text{diss}} = 0.1607048 \pm 0.0007717\, E_H$$

3.2 Analisi con metodi computazionali classici

Possiamo iniziare la nostra analisi con i metodi computazionali classici illustrati nell'introduzione nella sezione 1.2: FCI, UCCSD e MPTT.
A livello di programmazione si utilizza il *framework* PySCF[37] che contiene tutte le funzioni utili per uno studio classico[2].
La prima cosa da fare è indicare la struttura della molecola:

```
geometry = "H .0 .0 .0; H .0 .0 1.0; H .0 1.0 1.0"
mol = gto.M(atom=geometry,charge=1,spin=0,basis='sto-6g',
            symmetry=True,verbose=0)
```

Ad esempio nella prima delle due righe di codice appena scritte abbiamo indicato una molecola con tre atomi di idrogeno posizionati rispettivamente nello spazio (x,y,z) in (0,0,0), (0,0,1) e (0,1,1). Nella seconda riga abbiamo inizializzato la molecola indicandone, oltre alla geometria, anche carica, spin e base (in questo caso STO-6G).
I metodi che vogliamo andare a sondare sono HF, MPTT, FCI e CCSD. In PySCF bisogna per prima cosa calcolare la soluzione HF, poiché i metodi usati per gli altri approcci partono da tale soluzione.

```
mf  = scf.RHF(mol)
Ehf = mf.kernel()
```

Con la prima riga applichiamo il metodo di Hartree-Fock (in particolare RHF, come si può notare) e con la seconda "estraiamo" l'informazione sull'energia.
Per gli altri metodi:

```
mp2   = mp.MP2(mf)
e_mp2 = mp2.kernel()[0]
e_mp2 += Ehf

fci_h3 = fci.FCI(mf)
e_fci = fci_h3.kernel()[0]

ccsd_h3 = cc.CCSD(mf)
e_ccsd  = ccsd_h3.kernel()[0]
e_ccsd += Ehf
```

[2] In questa tesi cercherò di riportare le righe fondamentali di codice e di spiegarlo il più possibile. Necessariamente dovrò limitarmi a ciò che è fondamentale, ma tutti i codici utilizzati sono reperibili in forma integrale sulla pagina di GitHub del progetto[38].

Come si vede, per tutti e tre i metodi la soluzione non parte più dalla molecola stessa (mol), ma dalla soluzione col metodo di Hartree-Fock (mf). Si fa notare, inoltre, che la soluzione con la teoria di Møller-Plesset e la CCSD calcolano la differenza di energia da quella calcolata con HF.
A questo punto possiamo andare a variare la configurazione spaziale degli ioni della molecola per vedere quale distanza minimizza l'energia. Dobbiamo definire bene la distanza e quale grandezza andiamo a calcolare: ad esempio potremmo far variare in contemporanea tutti i lati di un triangolo equilatero; così facendo avremmo sì la giusta energia minima, ma l'energia di dissociazione (calcolabile come differenza fra energia a grande distanza ed energia minima) sarebbe scorretta. Infatti, l'energia di dissociazione di H_3^+ è calcolata con 3.1.0.2.
Potremmo, dunque, fissare due atomi di idrogeno alla distanza nota di H_2 e spostare unicamente il terzo a varie distanze. Così, però, avremmo corretta l'energia a grandi distanze, ma scorretta l'energia minima.
Per ora considereremo la variazione del lato di un triangolo equilatero, tenendo comunque a mente il "problema" dell'energia all'infinito[3].
Otteniamo il grafico in figura 3.2.1.

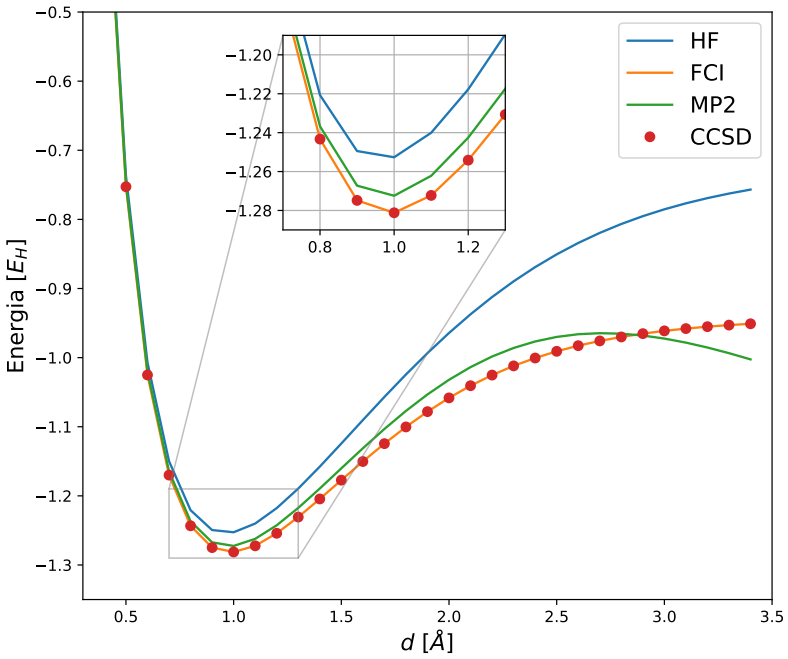


Figura 3.2.1: Comparazione fra metodi classici

[3]Evidentemente per calcolare l'energia di dissociazione dovremo poi applicare entrambi i metodi e prendere la differenza fra l'energia all'infinito del secondo e l'energia minima del primo.

Si vedono chiaramente due elementi già citati nell'introduzione (1.2):

- Il metodo di Hartree-Fock non è, evidentemente, sufficiente. In particolare l'errore che introduce è maggiore con l'aumentare della distanza.
- Come atteso i metodi FCI e CCSD coincidono per H_3^+: essendoci solo due elettroni, infatti, le eccitazioni tenute in considerazione da CCSD sono tutte quelle possibili

L'energia minima calcolata con il metodo FCI risulta[4]:

$$E_{\min}|_{FCI} = -1.280\ E_H$$

Per il grafico in figura 3.2.1 ho utilizzato la base STO-6G, ma nulla mi vieta di utilizzare una base migliore che, comunque, rimane gestibile per una molecola semplice come H_3^+. In sezione 1.1.2 abbiamo evidenziato come le basi minimali siano fondamentalmente insufficienti per qualsiasi calcolo, anche per molecole molto semplici come H_3^+; ci aspettiamo, dunque, un miglioramento sostanziale dei risultati all'aumentare della complessità della base.
Nel grafico in figura 3.2.2 sono riportati i risultati per FCI con diverse basi.

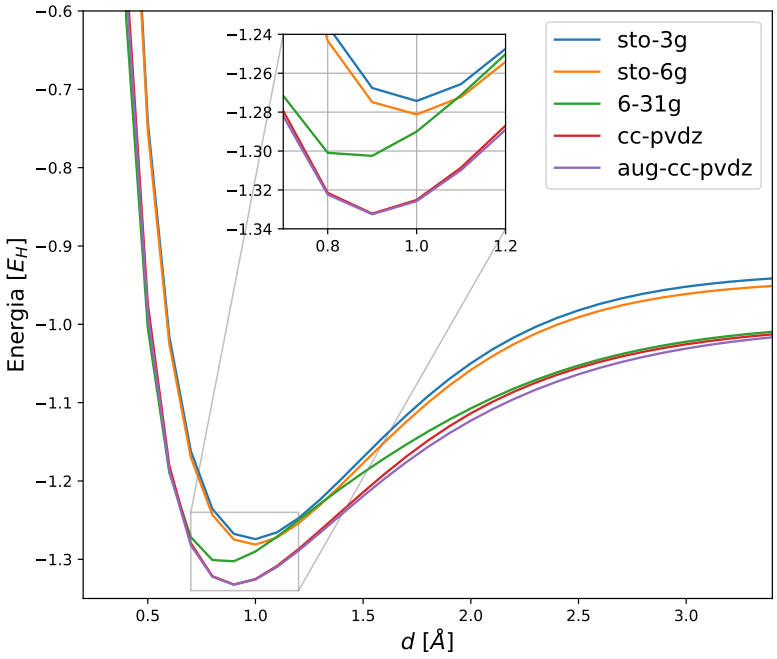


Figura 3.2.2: Comparazione fra basi diverse (FCI)

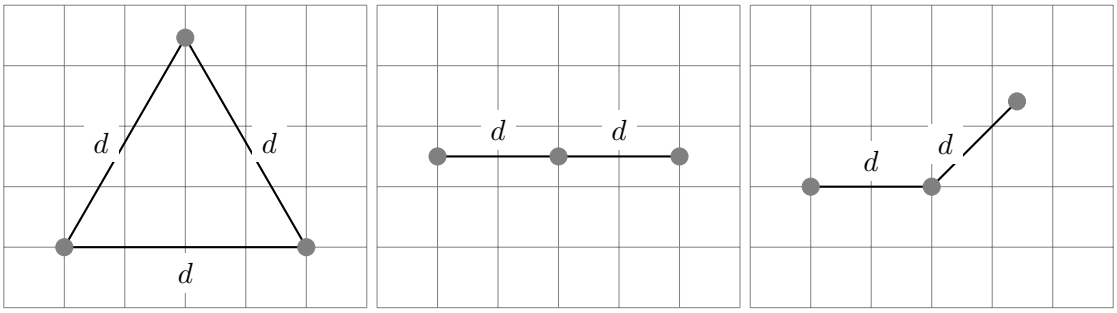
[4]Il risultato è privo di errore statistico perché ricavato da calcoli esatti, ma esiste una componente di errore sistematico dato dalla limitatezza della base.

Si vede chiaramente la differenza fra metodo e metodo. In particolare è ben visibile il distacco, in termini di risultati intorno al minimo, fra basi minimali (fra loro simili), la base 6-31G e le basi correlate (anche queste ultime fra loro simili). Come si può vedere, la scelta di una base di dimensioni maggiori porta a delle differenze di tipo qualitativo nella forma della superficie energetica. In particolare, estendere la base porta ad uno spostamento marcato della posizione di equilibrio della molecola, che risulta sovrastimata nel caso di basi minimali. Si nota, poi, che 6-31G e le *correlated basis* hanno medesimo comportamento "all'infinito".

3.2.1 Altre possibili geometrie molecolari

Come detto in sezione 3.1, la scelta di una geometria triangolare per una molecola come H_3^+ non é scontata.

Partendo senza alcuna conoscenza a priori della configurazione del catione idrogenonio, possiamo identificare tre differenti geometrie: una è quella triangolare già mostrata, una è una geometria lineare e l'ultima geometria è relativa a una configurazione *bended*[5]:



geometria triangolare geometria lineare geometria *bended*

Variando la distanza d ed eseguendo i calcoli con FCI possiamo verificare quale geometria è la più efficacie, ovvero quale geometria permette di raggiungere un'energia minore. Otteniamo il risultato in figura 3.2.3 dove possiamo chiaramente vedere che la geometria corretta è, in effetti, quella triangolare.

[5]Per la configurazione *bended* sarebbe opportuno far variare l'angolo liberamente; tuttavia in questo caso ho considerato, per semplicità, un angolo fisso a 45°.

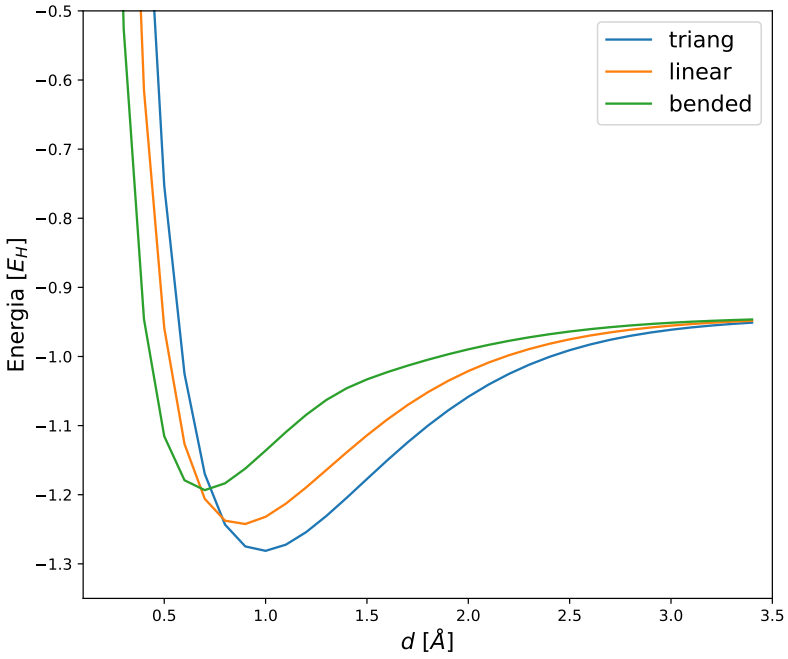


Figura 3.2.3: Confronto fra diverse geometrie molecolari (è stato usato il metodo FCI con una base STO-6G)

3.3 Simulazioni con VQE

3.3.1 Simulazioni esatte - statevector simulator

Procediamo, dunque, nello scrivere un programma che usi il *Variational Quantum Eigensolver*.

Dobbiamo definire anche qui una molecola, questa volta con PySCFDriver (classe di Qiskit che fa da ponte fra i propri metodi di *quantum computing* e la libreria di chimica PySCF):

```
driver = PySCFDriver(atom = geometry,
                     unit=UnitsType.ANGSTROM, spin = spin,
                     charge=charge, basis='sto-6g',
                     hf_method=HFMethodType.RHF)

molecule = driver.run()
```

Ricaviamo l'Hamiltoniano corrispondente al sistema e lo mappiamo in somme di operatori di Pauli (qubitOp) [6]:

```
core = Hamiltonian(transformation=TransformationType.FULL,
                   qubit_mapping=QubitMappingType.PARITY,
                   two_qubit_reduction=True, freeze_core=False)

qubitOp, aux_ops = core.run(molecule)
```

Calcoliamo lo stato iniziale del sistema in approssimazione di Hartree-Fock:

```
num_spin_orbitals=molecule.num_orbitals*2
num_particles= molecule.num_alpha + molecule.num_beta

initial_state = HartreeFock(num_spin_orbitals=num_spin_orbitals,
                            num_particles= num_particles,
                            qubit_mapping=qubit_mapping,
                            two_qubit_reduction=two_qubit_reduction)
```

A questo punto ricaviamo la forma variazionale, in questo caso una qUCCSD con parametri iniziali nulli, e inizializziamo l'algoritmo:

```
var_form = UCCSD(num_orbitals=num_spin_orbitals,
                 num_particles=num_particles,
                 initial_state=initial_state,
                 qubit_mapping=qubit_mapping,
                 two_qubit_reduction=two_qubit_reduction,
                 z2_symmetries=None)

init_parm = np.zeros(var_form.num_parameters)

vqe = VQE(qubitOp, var_form, optimizer, initial_point=init_parm)
```

Per avviare l'algoritmo dobbiamo scegliere un backend: utilizziamo, per questo primo tentativo, lo *statevector simulator*. Quest'ultimo é un simulatore esatto, che calcolerà la matrice unitaria corrispondente al circuito e lo applicherà al vettore che descrive lo stato dei qubit, prima di misurare il valore di aspettazione dell'Hamiltoniano su di esso. Dobbiamo anche ricordarci di sommare al termine di energia ricavato con VQE, il termine di repulsione internucleare:

```
backend = Aer.get_backend("statevector_simulator")
vqe_result_tot = vqe.run(backend)
shift = molecule.nuclear_repulsion_energy
energy =  np.real(vqe_result_tot['eigenvalue'] + shift)
```

Si è utilizzata, per i calcoli con VQE, la base minimale STO-6G con l'isomorfismo di parità e il *two_qubit_reduction* per ridurre al minimo il numero di qubit. Nella tabella 3.1 è riportato, nel caso di H3+, il numero di qubit (non ridotti) al variare della base: si vede chiaramente come l'hardware NISQ limiti le performance in modo determinante.

[6] Tramite aux_ops possiamo scegliere di calcolare anche altri operatori oltre all'Hamiltoniana, come, ad esempio, lo spin totale del sistema.

Base	Numero di qubit
sto-3g	6
sto-6g	6
6-31g	12
cc-pvdz	30

Tabella 3.1: Numero di qubit necessari per diverse basi (senza alcuna riduzione)

Per esser certo che VQE stia funzionando a dovere grafico i suoi risultati con quelli ottenuti precedentemente coi metodi classici (figura 3.2.1), ometto CCSD dal momento che coincide con FCI:

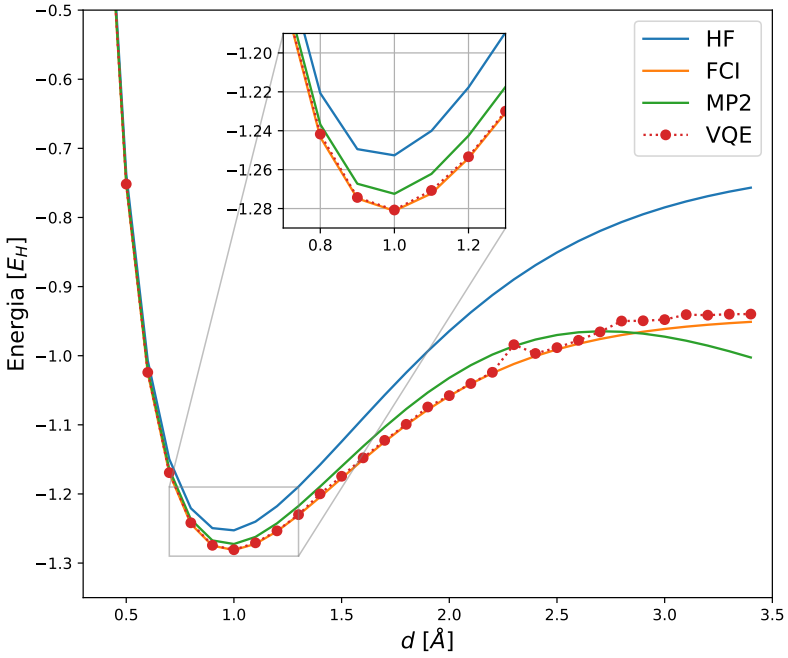


Figura 3.3.1: Comparazione fra VQE e metodi classici

Possiamo vedere nel grafico in figura 3.3.1 che il VQE ha risultati perfettamente in linea con la soluzione esatta. Specifichiamo di nuovo, tuttavia, che questa "soluzione esatta" è tale per una base minimale insufficiente, come visto in figura 3.2.2.

3.3.2 Noise Models

Il prossimo passo è, dunque, quello di eseguire VQE su QASM simulator, con l'aggiunta di un modello di rumore. In particolare, in questa sezione, paragoneremo i modelli di

rumore di due differenti computer: *ibmq_santiago* e *ibmq_16_melbourne*.

```
backend = Aer.get_backend("qasm_simulator")
device = provider.get_backend("ibmq_santiago")

coupling_map = device.configuration().coupling_map
noise_model = NoiseModel.from_backend(device)
basis_gates = noise_model.basis_gates

quantum_instance = QuantumInstance(backend=backend,
                    coupling_map=coupling_map,
                    noise_model=noise_model,shots=8000)

result = vqe.run(quantum_instance)
```

Si noti che non si importa solo il *noise model* (che contiene il rate di errori dei singoli gate e delle misure), ma anche la *coupling map* (ovvero le informazioni di connettività fra qubit) e i *basis_gates* (ovvero il set universale di gate fisici implementati).

In figura 3.3.2 sono riportati i risultati col rumore ricavato da *ibmq_santiago* (un *device* di ultima generazione) e *ibmq_16_melbourne* (*device* più vecchio, con più qubit disponibili, ma molto rumoroso) comparati al risultato del simulatore precedentemente analizzato.

Notiamo immediatamente l'effetto distruttivo del rumore sul profilo qualitativo della curva. Non solo la alza, aumentando l'errore nel calcolo del minimo, ma la distorce qualitativamente. Si nota, infatti, che la curva di *ibmq_16_melbourne* è strettamente decrescente e non vi è alcun minimo, mentre per *ibmq_santiago* il minimo è appena evidente. Ciò è chiaramente un grande problema: immaginando di studiare una molecola non perfettamente conosciuta rischieremmo di non vedere nemmeno il legame, classificando la struttura come instabile.

Dobbiamo anche considerare che un modello di rumore rappresenta la migliore esecuzione possibile sul corrispondente computer, quindi in un calcolo effettivo (non su un simulatore) la curva peggiorerà ancora.

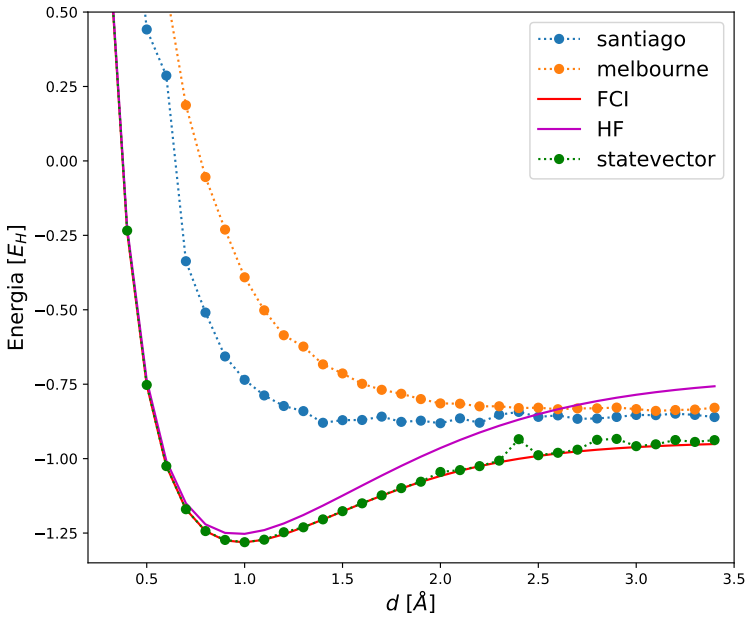


Figura 3.3.2: qUCCSD con Noise model (*ibmq_santiago* e *ibmq_16_melbourne*)

Calcolo dell'errore

I risultati di VQE sono stati considerati fino ad ora come privi di errore: in effetti è così che la funzione VQE di Qiskit ci restituisce l'energia trovata. Sarebbe tuttavia interessante avere una deviazione standard o, comunque, un'indicazione sulla possibile incertezza dei risultati.
Possiamo provare, in prima istanza, ad accedere ai dati "nascosti" dell'ottimizzatore tramite una *callback function*. Possiamo procedere in questo modo:

```
counts = []
values = []
std_var = []

def store_intermediate_result(eval_count, parameters, mean, std):
        counts.append(eval_count)
        values.append(mean)
        std_var.append(std)

vqe = VQE(qubitOp, var_form, optimizer, initial_point=init_param,
        callback=store_intermediate_result)

vqe_result_tot = vqe.run(quantum_instance)
energy = np.real(vqe_result_tot['eigenvalue'] + shift)
error = std_var[std_var.len() - 1]
```

La funzione *callback* viene chiamata ogni volta che l'ottimizzatore effettua un calcolo del valore della funzione e ha come parametri il numero di valutazioni già eseguite, il valore della funzione e l'errore relativo. Per come é definita, registra queste informazioni in vettori a cui si potrà accedere successivamente.
Il problema fondamentale di questo metodo, però, è che vengono considerate solo incertezze statistiche, mentre quelle di tipo sistematico non sono sondabili. Le incertezze statistiche per una simulazione di VQE con *noise model* risultano essere nell'ordine di $10^{-3}\,E_H$, ma abbiamo visto chiaramente (ad esempio in figura 3.3.2) che gli errori introdotti dal rumore sono molto maggiori. Questo perché il rumore, in un *quantum computer*, introduce un'incertezza sistematica e non statistica.
Potrei anche riportare, nei grafici seguenti, gli errori ricavati dalla funzione di *callback*, ma non sarebbero rappresentativi dell'errore totale. Per questo motivo non riporteremo alcun errore, avendo chiarito che debbano essere considerati dei grandi errori sistematici e dei piccoli errori statistici.

3.3.3 Scelta dell'ansatz

La figura 3.3.3 rappresenta il circuito per una qUCCSD inizializzata con parametri casuali.
Interessante, per prima cosa, guardare la prima colonna (a *depth*=1): essa rappresenta lo stato iniziale con l'approssimazione HF. I primi due qubit, infatti, rappresentanti due stati di spin opposto (rispettivamente α e β), descrivono stati occupati dopo l'applicazione di un NOT gate.
Il gate applicato al terzo qubit, invece, non rappresenta più lo stato iniziale del sistema, ma fa già parte degli operatori di eccitazione descritti alla sezione 2.2.2.
Come evidente, si tratta di un circuito estremamente lungo: è questo il problema principale che abbiamo incontrato nella precedente sezione.
Quando realizziamo un circuito, infatti, dobbiamo tenere conto degli errori che i gate introducono. Come già accennato in sezione 2.3.1, i gate più rumorosi sono i CNOT e il numero di essi è determinante per la riuscita di un programma. In generale non vogliamo superare la decina di CNOT. L'ansatz qUCCSD per H3+ ne conta 171. Dunque con lo *statevector simulator* non avremo alcun problema, ma abbiamo visto che introdurre rumore deteriora i risultati. Su un computer reale, poi, questi possono solo peggiorare in quanto il *noise model* è una semplificazione che non tiene conto di una serie di altri parametri variabili.
Dunque, malgrado l'efficienza teorica di qUCCSD, nell'era NISQ non possiamo far altro che andare a cercare altre forme variazionali più brevi come illustrato nella sezione 2.2.2.

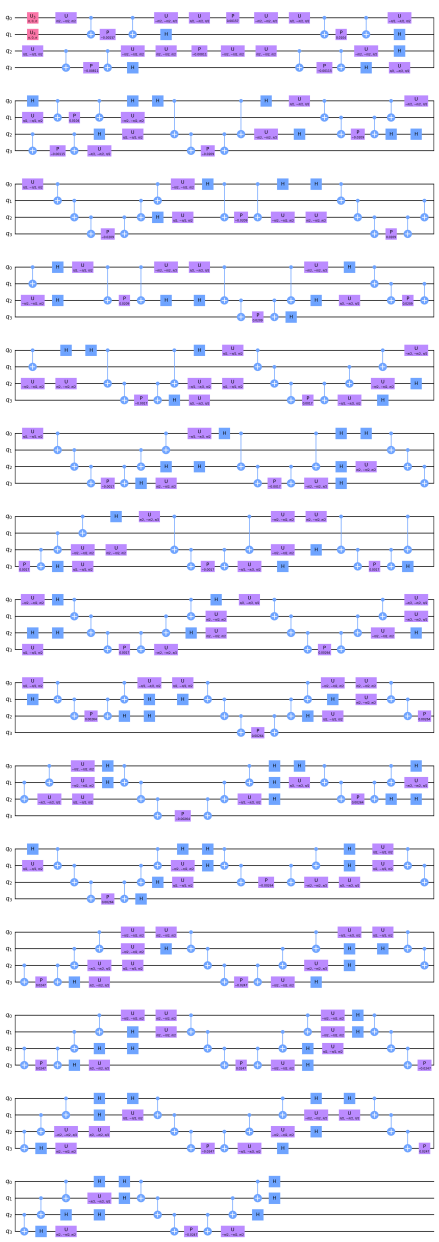


Figura 3.3.3: qUCCSD Ansatz

Possiamo implementare l'ansatz hardware efficient rotazionale generico usando la classe di Qiskit "TwoLocal":

```
var_form = TwoLocal(rotation_blocks='ry', entanglement_blocks='cx',
                    num_qubits=4,
                    reps=3,
                    initial_state=initial_state,
                    entanglement='linear')
```

TwoLocal è una classe di Qiskit molto flessibile e permette di variare sia i "gate rotazionali" (ad esempio avremmo potuto mettere ['ry','rz']) che i gate di entaglement (in sezione 2.1.1 abbiamo illustrato solo CNOT, ma in modo analogo si possono definire altri gate di controllo come, ad esempio, CZ). Il numero di *reps*, ripetizioni, è uguale a 3 (il valore predefinito), ma è possibile ridurlo o aumentarlo a piacimento. Ovviamente una maggiore profondità del circuito permette una maggiore generalità dell'ansatz, rendendo peró più difficile il processo di minimizzazione.

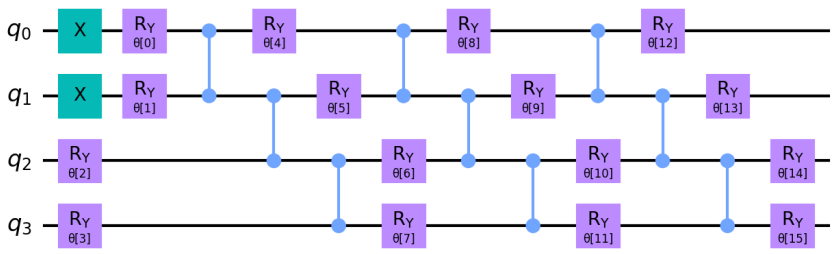


Figura 3.3.4: Hardware efficient Ansatz (R_y)

Per SO(4), invece, dobbiamo costruire da zero la forma variazionale poiché questa non è implementata in Qiskit. Partiamo costruendo il gate per due qubit (che in sezione 2.2.2 abbiamo chiamato, appunto, SO(4)) visibile in figura 3.3.5.

```
def add_unitary_gate(circuit,qubit1,qubit2,params,p0):
    circuit.s(qubit1)
    circuit.s(qubit2)
    circuit.h(qubit2)
    circuit.cx(qubit2,qubit1)
    circuit.u3(params[p0],params[p0+1],params[p0+2],qubit1); p0 += 3
    circuit.u3(params[p0],params[p0+1],params[p0+2],qubit2); p0 += 3
    circuit.cx(qubit2,qubit1)
    circuit.h(qubit2)
    circuit.sdg(qubit1)
    circuit.sdg(qubit2)
```

Come si può notare, stiamo lasciando completamente liberi i parametri che dovranno infatti essere variati dall'ottimizzatore classico.
A questo punto, partendo dalla *abstract class* "VariationalForm", possiamo definire un nostro ansatz.

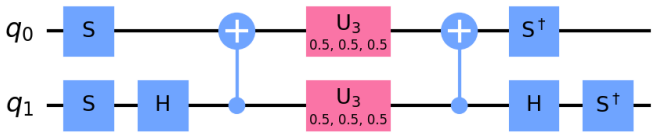


Figura 3.3.5: SO(4) gate, i parametri sono stati inizializzati tutti a 0.5 arbitrariamente

```
from qiskit.aqua.components.variational_forms import VariationalForm

class SO_04(VariationalForm):  #definizione della classe
    def __init__(self, numqubits):
        self._num_qubits = numqubits
        self._num_parameters = 6*(numqubits-1)

    def construct_circuit(self, parameters):
        q = QuantumRegister(self._num_qubits, name='q')
        circ = QuantumCircuit(q)

        #initial state
        circ.x(0)
        circ.x(1)
        #definizione del circuito
        i = 0
        n = 0

        while i + 1 < self._num_qubits:
            add_unitary_gate(circ, i, i+1, parameters, 6*n)
            n = n +1
            i = i + 2

        i = 1

        while i + 1 <  self._num_qubits:
            add_unitary_gate(circ, i, i+1, parameters, 6*n)
            n = n +1
            i = i + 2

        return circ

    @property
    def num_parameters(self):
        return self._num_parameters

var_form = SO_04(4)
```

Nel complesso, per un circuito a quattro qubit, la forma variazionale (con stato iniziale incluso) risulta quella in figura 3.3.6.

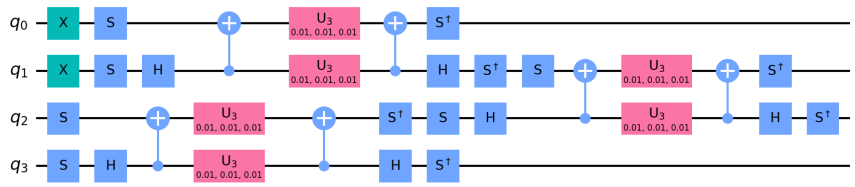


Figura 3.3.6: SO(4) ansatz

Per prima cosa si prova a simulare i tre diversi ansatz con lo *statevector simulator*. Un "buon risultato" si trova tra la curva HF (da intendere come il peggior risultato che possiamo ottenere coi metodi classici) e FCI (il migliore); in effetti, intorno al minimo, vediamo in figura 3.3.7[7] esattamente questo tipo di risultato, ma è ben chiaro che qualcosa non funziona a distanze maggiori di $\approx 1.5\,\text{Å}$.

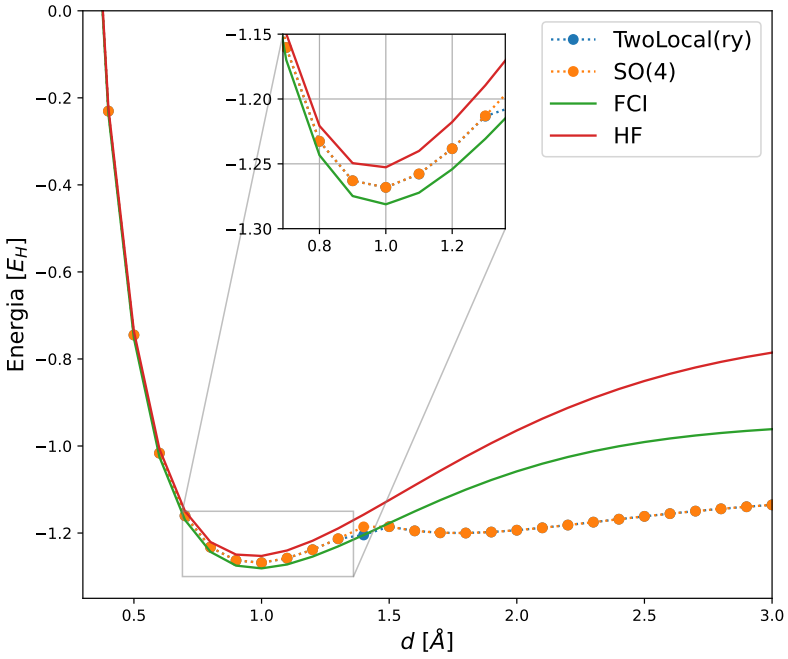


Figura 3.3.7: Ansatz efficienti, simulati con *statevector simulator*

Possiamo modificare leggermente il nostro codice in modo che l'algoritmo non calcoli solo il valore di aspettazione dell'Hamiltoniana (che rimane, in ogni caso, la funzione

[7] I due ansatz danno risultati generalmente differenti solo alla terza cifra decimale: in questo grafico, dunque, i marker sono quasi sempre sovrapposti.

costo usata dall'ottimizzatore classico), ma anche il numero di particelle[8].

```
qubitOp, aux_ops = core.run(molecule)

aux_ops = aux_ops[:3]
aux_ops.append(qubitOp)

...

vqe = VQE(..., aux_operators = aux_ops)
result = vqe.run(backend)

name = ['numero di particelle','spin2','spin_z']
for i in range(3):
    print("\t operatore ",name[i],
        result['aux_operator_eigenvalues'][i,0])
```

Si può vedere, in tal modo, che da una certa distanza[9] gli ansatz *hardware efficient* "trovano" una configurazione a quattro elettroni ad energia minore di quella corretta per H_+^3 (entrambi gli ansatz efficienti hanno lo stesso problema). Questo è un problema consueto per gli ansatz come TwoLocal o SO(4) poiché entrambi non partono da ragionamenti chimico-fisici, al contrario di qUCCSD.
Una possibile soluzione a questo problema, che però non è utilizzata in questa tesi, è quella di sostituire nel processo di minimizzazione l'Hamiltoniana con un operatore lagrangiano del tipo:

$$L = H - \lambda_1(\bar{N} - N) - \lambda_2(\bar{S}^2 - S^2) - \lambda_3(\bar{S}_z - S_z) \tag{3.3.3.1}$$

In questo modo, scegliendo adeguatamente i moltiplicatori di Lagrange (i coefficienti λ) è possibile spingere l'ottimizzatore a mantenere costante il numero di particelle e i valori di spin, senza costo computazionale aggiuntivo (vengono solo aggiunti degli operatori, e quindi saranno necessarie misure aggiuntive ma non comportano profondità maggiori del circuito).
In ogni caso questo non è l'unico problema che incontriamo nell'utilizzo di questi ansatz: infatti, intorno ai valori energetici descritti da HF, gli ansatz *hardware efficient* hanno spesso un gradiente nullo. Si tratta di un minimo locale che può ostacolare il processo di ottimizzazione.
Proprio per questo è inutile, anzi spesso dannoso, usare come stato iniziale quello derivante dall'approssimazione HF e inizializzare i parametri a 0.
In ogni caso non ci deve preoccupare troppo questo problema di conservazione: la configurazione ionica che stiamo studiando, infatti, considera una geometria triangolare dove si allontanano man mano tutti gli atomi, ma la dissociazione di H_3^+ porta a una molecola di idrogeno e un atomo singolo. Sostanzialmente, dunque, come già detto in sezione 1.2, siamo interessati in questo momento solo all'energia a piccole distanze.

[8] Così scritto l'algoritmo calcola anche il valore dello spin

[9] Da notare, inoltre, che la distanza a cui la configurazione a 4 elettroni diventa la preferita non è costante, ma dipende in particolare dal valore dei parametri iniziali.

3.3.4 Misure con *Error Mitigation*

Come abbiamo visto precedentemente (sezione 2.1.4) è possibile implementare diverse tecniche di riduzione degli errori. In Qiskit possiamo facilmente implementare una correzione SPAMEM con una leggera modifica all'inizializzazione della quantum instance:

```
quantum_instance = QuantumInstance(backend=backend,
                   seed_simulator=seed,
                   seed_transpiler=seed, coupling_map=coupling_map,
                   noise_model=noise_model, shots=8000,
                   measurement_error_mitigation_cls=CompleteMeasFitter,
                   measurement_error_mitigation_shots=1000)
```

Per mostrare il miglioramento metto a grafico i risultati di VQE (base STO-6G, ansatz TwoLocal) con e senza correzione errori.

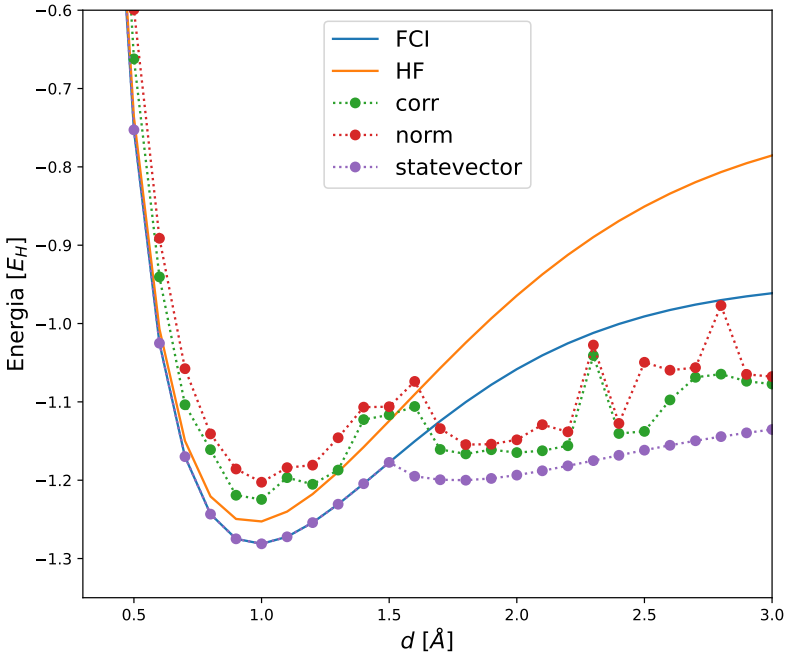


Figura 3.3.8: Confronto fra VQE con e senza correzione SPAMEM ("norm" sta per normale, dunque VQE senza correzioni, contrapposto a "corr" che ha correzioni)

Come si può vedere chiaramente, il grafico in figura 3.3.8 segue la curva del risultato dello statevector solo parzialmente: in vari punti il grafico è molto frastagliato e non rappresenta quello che ci aspettiamo. Con ogni probabilità il processo di ottimizzazione, a causa del rumore, non trova il minimo corretto dell'energia, ma si stabilizza su minimi locali che non c'entrano nulla col sistema reale. In particolare bisogna ricordare di nuovo

che TwoLocal è un ansatz *hardware agnostic* e non è tenuto ad avere alcuna coerenza col sistema fisico in analisi.
Abbiamo già visto in sezione 3.3.3 come sia presente una configurazione con un numero differente di elettroni che presenta un minimo nell'energia evidentemente più "largo" del minimo assoluto della configurazione a due elettroni. Aggiungendo nuovamente il calcolo degli operatori ausiliari nella definizione di VQE possiamo confermare che il problema è, come prima, nella conservazione del numero di particelle e, inoltre, nella conservazione dello spin.
Una possibile soluzione sarebbe modificare l'ottimizzatore ad hoc in modo da evitare un minimo locale, variando con cura sia la tolleranza che lo step con cui lavora. Alternativamente, come già accennato, è possibile cambiare operatore da misurare (i.e. l'hamiltoniano) in un nuovo operatore che "costringa" la conservazione del numero di particelle.
Tuttavia, possiamo ancora ricordarci che la misura che abbiamo adesso eseguito aveva lo scopo di trovare l'energia minima di H_3^+ (a 1 Å) che, nel grafico 3.3.8, è correttamente calcolata.
Possiamo, dunque, identificare l'energia minima[10]:

$$E_{min}|_{senza\ correzione} = -1.202\ E_H$$
$$E_{min}|_{con\ correzione} = -1.224\ E_H$$

Tornando alla discussione principale di questo paragrafo, possiamo vedere chiaramente come la correzione SPAMEM sia solo parzialmente efficiente. Come già detto non tiene in considerazione ogni tipo di errore e dunque non riporta perfettamente i valori misurati a quelli ricavati dalla simulazione senza rumore, ma c'è comunque un visibile miglioramento.

3.3.5 Calcolo dell'energia di dissociazione

Fino ad ora abbiamo eseguito simulazioni concentrandoci sull'energia minima alla distanza di legame di H_3^+, ma non dobbiamo dimenticarci che un altro obiettivo è il calcolo della sua energia di dissociazione.
Dobbiamo quindi modificare la geometria della nostra molecola, da:

```
dist = np.arange(0.2, 3.5, .1)
alt = np.sqrt(dist**2 - (dist/2)**2)

for i in range(len(dist)):
    geometry = "H .0 .0 .0;" +
               "H .0 .0 " + str(dist[i]) +
               "; H .0 " + str(alt[i]) + " " + str(dist[i]/2)
```

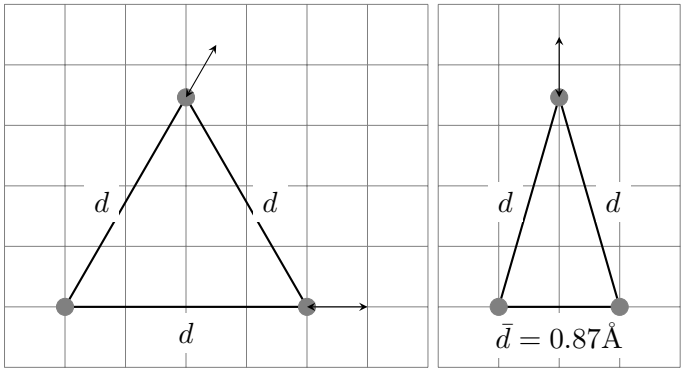
[10]Non sono riportati errori, ma come già detto in sezione 3.3.2 abbiamo sempre errori statistici molto piccoli ed errori sistematici molto grandi.

a:

```
dist = np.arange(0.2, 3.5, .1)

for i in range(len(dist)):
    geometry = "H .0 .0 .0;" +
               " H .0 .0 0.87;" +
               " H .0 " + str(dist[i]) + " 0.435"
```

Nel secondo caso teniamo "fissa" una molecola di H_2 alla distanza di legame nota ($d = 0.87$Å) e spostiamo il rimanente atomo di idrogeno. Nell'immagine successiva sono rappresentati i due diversi schemi:



In tal modo l'energia a distanze piccole sarà falsata e non interessante, ma otterremo i risultati che cerchiamo "all'infinito".

In figura 3.3.9 sono graficati i risultati per FCI, HF e VQE con TwoLocal con e senza rumore (base sto-6g, modello di rumore importato da *ibmq_santiago* e correzioni SPAMEM).

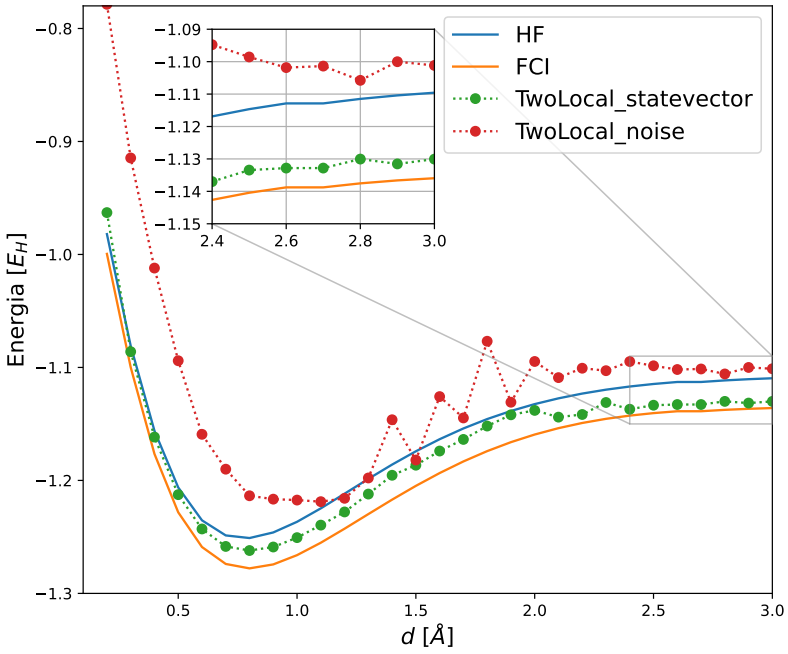


Figura 3.3.9: Calcoli (HF, FCI, VQE con e senza rumore) per l'energia di dissociazione

Possiamo, dunque, identificare approssimativamente i due risultati principali di FCI e VQE con rumore:

$$E_\infty|_{VQE} = -1.100 \pm 0.005\ E_H$$
$$E_\infty|_{FCI} = -1.135 \pm 0.005\ E_H$$

3.4 Misure su *Quantum Hardware*

Eseguiamo, infine, un'analisi completa dell'energia di dissociazione su *quantum hardware* importando il corretto *backend*:

```
provider = IBMQ.get_provider('ibm-q')
backend = provider.get_backend('ibmq_santiago')
hardware = QuantumInstance(backend=backend, shots=8000,
              measurement_error_mitigation_cls=CompleteMeasFitter,
              measurement_error_mitigation_shots=1000)
```

Per ridurre al minimo i calcoli necessari saranno mischiate le due configurazioni introdotte in sezione 3.3.5, alla distanza arbitraria di 2Å si passerà dalla prima configurazione alla seconda[11].
Si deve utilizzare necessariamente un ansatz *hardware efficient* e viene scelto TwoLocal (i risultati ottenuti con SO(4) sono perfettamente compatibili).
Ricordiamo ancora che l'utilizzo della base STO-6G é fortemente limitante, ma il meglio che possiamo implementare[12] e si considera, inoltre, una correzione SPAMEM come in sezione 3.3.4.
I risultati sono visibili in figura 3.4.1.

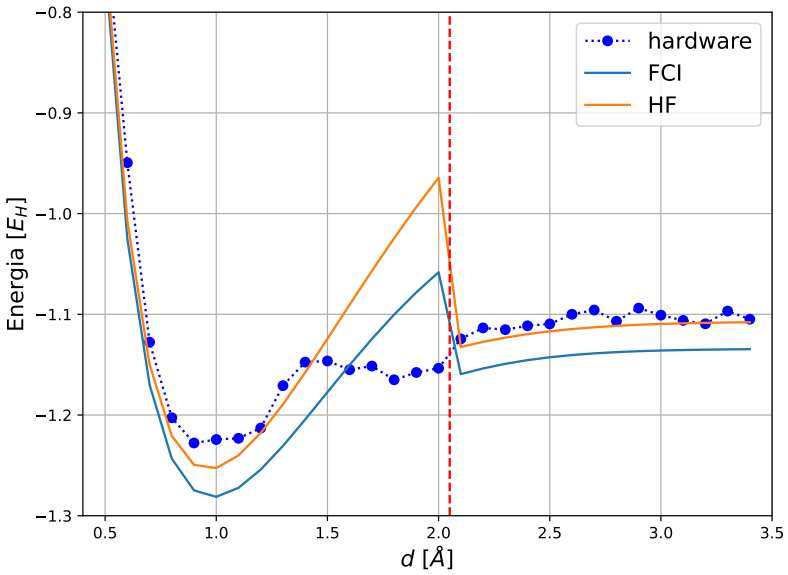


Figura 3.4.1: Misura su quantum hardware

Come ci aspettavamo dalle precedenti simulazioni, a 1.5Å i risultati di VQE si discostano dai valori corretti. Generalmente, invece, i risultati ricavati dalle misure su *hardware* sono leggermente superiori a quelli ottenuti con HF e rimangono compatibili con i valori simulati.

[11] Chiaramente il processo di dissociazione non avviene di colpo, ma ciò che considero in questo caso sono solo le energie "iniziali" e "finali".
[12] A meno di non sfruttare tecniche particolari per ridurre il numero il numero di qubit[39]

Capitolo 4

Conclusioni

4.1 Risultati

Il valore sperimentale tabulato dell'energia di dissociazione di H_3^+ è:

$$E|_{sperimentale} = 0.1607048 \pm 0.0007717 \; E_H$$

Valore compatibile con quanto abbiamo trovato col metodo computazionale classico della *Full Configuration Interaction* (illustrato in sezione 1.2) e la base correlata aug-cc-pvdz che abbiamo illustrato in sezione 1.1.2:

$$E|_{FCI(aug-cc-pvdz)} = 0.164 \pm 0.005 \; E_H$$

Dovendo limitarci a una base minimale STO-6G per il limite di 5 *qubit* nei *quantum computer* a nostra disposizione, tuttavia, il calcolo esatto FCI ha portato un valore inferiore:

$$E|_{FCI(sto-6g)} = 0.145 \pm 0.005 \; E_H$$

Questo è il valore migliore che il *Variational Quantum Eigensolver* può ottenere. In effetti ciò è confermato dal grafico 3.3.1, dove abbiamo comparato i risultati esatti con una simulazione di VQE senza rumore. La presenza del fenomeno della decoerenza nonché di altre fonti di rumore, tuttavia, introduce un considerevole errore sistematico.
Inizialmente abbiamo provato a utilizzare VQE con qUCCSD: una forma variazionale *problem inspired* estremamente efficiente dal punto di vista teorico, ma non utilizzabile in computer NISQ poichè comporta un *quantum circuit* troppo profondo.
Abbiamo dunque introdotto, in sezione 2.2.2, delle alternative *problem agnostic* costruite col preciso scopo di minimizzare la profondità dei rispettivi *quantum circuit*.
Combinando i risultati ottenuti nelle sezioni 3.3.4 e 3.3.5 per la forma variazionale rotazionale TwoLocal e considerando l'importazione di un *noise model* da *ibmq_santiago* e una correzione SPAMEM (che applica delle correzioni a una parte degli errori presenti, come illustrato alla sezione 2.1.4), otteniamo che l'energia di dissociazione di H_3^+ è:

$$E|_{VQE(noise)} = 0.124 \pm 0.005\ E_H$$

TwoLocal è un ansatz che si basa su rotazioni in spazio SU(2), abbiamo provato a utilizzare anche un ansatz che si basa su rotazioni in spazio SO(4) (illustrato in sezione 2.2.2) che ha portato a risultati sostanzialmente identici.
E, infine, abbiamo utilizzato VQE su *quantum hardware* ottenendo:

$$E|_{VQE(hardware)} = 0.119 \pm 0.005\ E_H$$

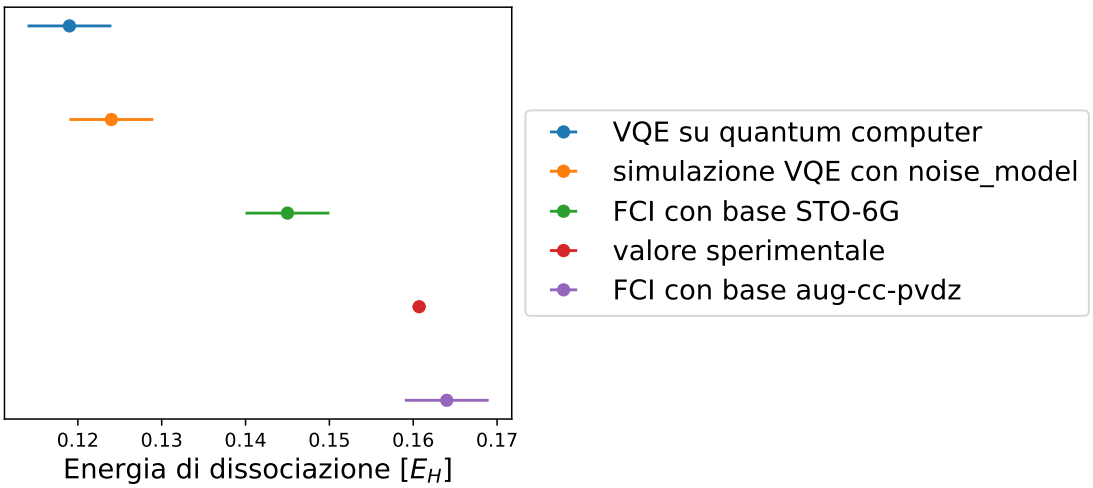


Figura 4.1.1: Distribuzione dei risultati

In figura 4.1.1 possiamo confrontare i risultati dei vari metodi e dobbiamo constatare che il valore ottenuto con VQE si discosta considerevolmente dal valore vero (di 7 deviazioni standard per la simulazione e 8 deviazioni standard per i risultati su *quantum hardware*), questo per due motivi principali:

- il limite nel numero di qubit che ha portato a un limite nella base;
- l'implementazione solo di una tecnica SPAMEM non è in grado di correggere tutti gli errori introdotti da un *quantum computer*.

4.2 Prospettive future

Il *Variational Quantum Eigensolver* è un algoritmo molto promettente in epoca NISQ, ma abbiamo visto come non sia uno strumento utilizzabile senza preoccupazioni. Questo algoritmo infatti condivide alcune limitazioni con i metodi variazionali classici. La scelta dell'ansatz, dell'optimizer *optimizer* e/o del punto iniziale sono tutte empiriche e non

univoche. Si rende quindi necessario uno studio approfondito del problema analizzato per capire quali scelte siano le migliori per lo specifico caso.
Sulla questione della forma variazionale abbiamo visto più volte come siano interessanti i risultati di un ansatz *problem inspired* e quali siano, però, i suoi problemi legati all'era NISQ. In questa tesi abbiamo esplorato i due estremi: siamo partiti da un ansatz totalmente "chimico" per arrivare a degli ansatz (SO(4) e TwoLocal) totalmente trascendenti dal problema reale. In realtà ci sono, evidentemente, numerose vie di mezzo che consentono di mantenere una profondità gestibile senza incorrere in quei problemi di mancata conservazione di quantità fisiche che abbiamo dovuto affrontare[40].
Inoltre, la letteratura più recente sul *Variational Quantum Eigensolver* propone numerose e varie soluzioni per rendere efficacie l'algoritmo anche sui computer NISQ con modifiche all'algoritmo stesso: si può vedere, per citarne alcune, il filone di ricerca che parte da QAOA[41] (*Quantum Approximate Optimization Algorithm*) o da ADAPT-VQE[42] (*Adaptive Derivative-Assembled Pseudo-Trotter ansatz Variational Quantum Eigensolver*). Entrambi questi algoritmi sono variazioni sullo stesso tema di VQE[1] e propongono valide soluzioni, almeno in alcuni casi, ai problemi evidenziati fino ad ora.
Tra i problemi che ho affrontato in questa tesi, poi, non tutti possono essere ricondotti a problemi intrinseci dell'algoritmo: ad esempio abbiamo visto che siamo limitati all'utilizzo di una base minimale STO-6G per il ridotto numero di qubit disponibili. Una recente soluzione[43] propone di "liberarsi" delle basi di funzioni per utilizzare, invece, un metodo variabile di caso in caso che ottimizzi il numero di qubit necessari a partire sia da HF che da MP2.
Nel finire di questa tesi vorrei discutere di un ultimo concetto: la *quantum supremacy* o *quantum advantage*[2]. Con questi termini si intende il momento in cui, effettivamente, un *quantum computer* sarà in grado di svolgere dei compiti non risolvibili da un computer classico[3]. Google nel 2019 ha pubblicato un articolo dimostrando di avere raggiunto la *quantum supremacy*[44]. Tale lavoro è un po' dibattuto e la stessa IBM ha subito cercato di mostrare come il compito risolto dal *quantum computer* di Google non fosse totalmente irrisolvibile da processori classici e che quindi fosse stata raggiunto solo un *advantage*[45]. In ogni caso il compito computazionale eseguito da Google non aveva alcuna utilità pratica, ma solo una valenza "accademica".
Sul fronte relativo al raggiungimento di una *quantum supremacy* dall'utilità pratica, il *Variational Quantum Eigensolver* e i suoi derivati sono generalmente considerati come gli algoritmi più promettenti. Le applicazioni sono estremamente interessanti e vanno dallo studio di molecole (o composti più complessi) utili nel campo delle Scienze dei Materiali e non solo, allo studio di sistemi di Fisica Nucleare o Subnucleare dove un *quantum computer* può evidentemente avere dei vantaggi rispetto ai calcolatori classici. Infine, non siamo neanche limitati a sistemi quantistici, ma possiamo utilizzare algoritmi

[1] In realtà QAOA è un algoritmo più indipendente da VQE di quanto non lo sia ADAPT-VQE, ciononostante si può vedere come forma più generale di VQE stesso.

[2] Supremazia quantistica o vantaggio quantistico.

[3] La differenza fra i termini non è ben definita, ma possiamo dire che l'*advantage* corrisponde alla soluzione di problemi in tempo significativamente minore che su un computer tradizionale, la *supremacy* alla soluzione di problemi irrisolvibili classicamente.

simili a VQE (ad esempio QAOA) per problemi di ottimizzazione classica o di *Machine Learning*.

Nella caccia al tesoro della computazione quantistica, in cui il tanto agognato forziere è la *quantum supremacy* a breve termine, la mappa da utilizzare potrebbe essere rappresentata proprio dai *Variational Quantum Algorithms*.

Riferimenti bibliografici

[1] Francesco Tacchino et al. «Quantum Computers as Universal Quantum Simulators: State-of-the-Art and Perspectives». In: *Advanced Quantum Technologies* 3.3 (2020), p. 1900052. ISSN: 2511-9044. DOI: 10.1002/qute.201900052. arXiv: 1907.03505.

[2] Richard P Feynman. *Simulating Physics with Computers*. Rapp. tecn. 6. 1982.

[3] David P. DiVincenzo e IBM. «The Physical Implementation of Quantum Computation». In: *Fortschritte der Physik* 48.9-11 (feb. 2000), pp. 771–783. arXiv: 0002077 [quant-ph]. URL: http://arxiv.org/abs/quant-ph/0002077.

[4] Shi-Yao Hou et al. *SpinQ Gemini: a desktop quantum computer for education and research*. 2021. arXiv: 2101.10017 [quant-ph].

[5] Lov K. Grover. «A fast quantum mechanical algorithm for database search». In: *Proceedings of the Annual ACM Symposium on Theory of Computing* Part F129452 (mag. 1996), pp. 212–219. arXiv: 9605043 [quant-ph]. URL: http://arxiv.org/abs/quant-ph/9605043.

[6] Peter W. Shor. «Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer». In: *SIAM Journal on Computing* 26.5 (ago. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/s0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172.

[7] Kishor Bharti et al. «Noisy intermediate-scale quantum (NISQ) algorithms». In: (2021), pp. 1–82. arXiv: 2101.08448. URL: http://arxiv.org/abs/2101.08448.

[8] Alberto Peruzzo et al. «A variational eigenvalue solver on a photonic quantum processor». In: *Nature Communications* (2014). DOI: 10.1038/ncomms5213. URL: www.nature.com/naturecommunications.

[9] Stefano Barison. «Study of nitrogen-copper compounds by quantum simulation algorithms». Tesi di laurea mag. Univeristà degli studi di Milano, 2020.

[10] Attila Szabo e Neil S. Ostlund. *Modern Quantum Chemistry*. New York: Dover Publications, INC., 1947. ISBN: 0-486-69186-1.

[11] Ernest R. Davidson e David Feller. «Basis Set Selection for Molecular Calculations». In: *Chemical Reviews* 86 (1986), pp. 681–696.

[12] Errol G. Lewars. *Computational Chemistry*. Seconda edizione. Springer, 2011. ISBN: 9789048138609.

[13] Christine Corbett Moran. *Mastering Quantum Computing with IBM QX*. 2019. ISBN: 1789136431.

[14] *Accessing Higher Energy States*. URL: https://qiskit.org/textbook/ch-quantum-hardware/accessing_higher_energy_states.html.

[15] Elias Fernandez-Combarro Alvarez. «A practical introduction to quantum computing: from qubits to quantum machine learning and beyond.» In: (nov. 2020). URL: https://indico.cern.ch/event/970903/.

[16] Sam McArdle et al. «Quantum computational chemistry». In: *Reviews of Modern Physics* 92.1 (2020). ISSN: 15390756. DOI: 10.1103/RevModPhys.92.015003. arXiv: 1808.10402.

[17] Sergey Bravyi et al. «Tapering off qubits to simulate fermionic Hamiltonians». In: 1 (2017), pp. 1–15. arXiv: 1701.08213. URL: http://arxiv.org/abs/1701.08213.

[18] Filip B Maciejewski et al. *Modeling and mitigation of cross-talk effects in readout noise with applications to the Quantum Approximate Optimization Algorithm*. Rapp. tecn. DOI: 10.22331/q-2021-06-01-464. arXiv: 2101.02331v3.

[19] Austin G. Fowler et al. «Surface codes: Towards practical large-scale quantum computation». In: *Physical Review A - Atomic, Molecular, and Optical Physics* 86.3 (ago. 2012). DOI: 10.1103/PhysRevA.86.032324. arXiv: 1208.0928. URL: http://arxiv.org/abs/1208.0928%20http://dx.doi.org/10.1103/PhysRevA.86.032324.

[20] Jakob M Günther et al. *Improving readout in quantum simulations with repetition codes*. Rapp. tecn. arXiv: 2105.13377v1.

[21] Manpreet Singh Jattana et al. «General error mitigation for quantum circuits». In: *Quantum Information Processing* 19 (2020), p. 414. DOI: 10.1007/s11128-020-02913-0. URL: https://doi.org/10.1007/s11128-020-02913-0.

[22] V. Armaos et al. «Computational chemistry on quantum computers: Ground state estimation». In: *Applied Physics A: Materials Science and Processing* 126.8 (2020), pp. 1–6. ISSN: 14320630. DOI: 10.1007/s00339-020-03755-4. arXiv: 1907.00362.

[23] P J J O'malley et al. *Scalable Quantum Simulation of Molecular Energies*. Rapp. tecn. 2017. arXiv: 1512.06860v2.

[24] Jarrod R Mcclean et al. *The theory of variational hybrid quantum-classical algorithms*. Rapp. tecn. arXiv: 1509.04279v1.

[25] M. Cerezo et al. «Variational Quantum Algorithms». In: (2020), pp. 1–29. arXiv: 2012.09265. URL: http://arxiv.org/abs/2012.09265.

[26] Panagiotis Kl Barkoutsos et al. «Quantum algorithms for electronic structure calculations: Particle-hole Hamiltonian and optimized wave-function expansions». In: *Physical Review A* 98.2 (2018), pp. 1–14. ISSN: 24699934. DOI: 10.1103/PhysRevA.98.022322. arXiv: 1805.04340.

[27] Joonho Lee et al. ≪Generalized Unitary Coupled Cluster Wave functions for Quantum Computation≫. In: *Journal of Chemical Theory and Computation* 15.1 (2019), pp. 311–324. ISSN: 15499626. DOI: 10.1021/acs.jctc.8b01004. arXiv: 1810.02327.

[28] Abhinav Kandala et al. ≪Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets≫. In: *Nature* 549.7671 (2017), pp. 242–246. ISSN: 14764687. DOI: 10.1038/nature23879. arXiv: 1704.05018.

[29] Andrew W Cross et al. *Validating quantum computers using randomized model circuits*. Rapp. tecn. 2019. arXiv: 1811.12926v2.

[30] Ludwik Adamowicz e Michele Pavanello. ≪Progress in calculating the potential energy surface of H3+≫. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 370.1978 (2012), pp. 5001–5013. ISSN: 1364503X. DOI: 10.1098/rsta.2012.0101.

[31] David Stevenson e Joseph Hirschfelder. ≪The Structure of H3, H3+, and of H3–. IV≫. In: *Journal of Chemical Physics* 5.12 (1937), pp. 933–940. ISSN: 10897690. DOI: 10.1063/1.1749966.

[32] Jonathan Tennyson. ≪Spectroscopy of H3+: Planets, chaos and the Universe≫. In: *Reports on Progress in Physics* 58.4 (1995), pp. 421–476. ISSN: 00344885. DOI: 10.1088/0034-4885/58/4/002.

[33] A. V. Turbiner e J. C. Lopez Vieyra. ≪Ground state of the H3+ molecular ion: Physics behind≫. In: *Journal of Physical Chemistry A* 117.39 (2013), pp. 10119–10128. ISSN: 10895639. DOI: 10.1021/jp401439c. arXiv: 1212.4552.

[34] Ralph G. Pearson. ≪Energy of H3+ and H3 by the method of molecular orbitals≫. In: *The Journal of Chemical Physics* 16.5 (1948), pp. 502–504. ISSN: 00219606. DOI: 10.1063/1.1746924.

[35] Michele Pavanello et al. ≪New more accurate calculations of the ground state potential energy surface of H3+≫. In: *Journal of Chemical Physics* 130.7 (2009), pp. 1–6. ISSN: 00219606. DOI: 10.1063/1.3077193.

[36] P. C. Cosby e H. Helm. ≪Experimental determination of the H3+ bond dissociation energy≫. In: *Chemical Physics Letters* 152.1 (nov. 1988), pp. 71–74. ISSN: 00092614. DOI: 10.1016/0009-2614(88)87330-5.

[37] Qiming Sun et al. ≪PySCF: the Python-based simulations of chemistry framework≫. In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 8.1 (gen. 2018), e1340. ISSN: 1759-0884. DOI: 10.1002/WCMS.1340. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/wcms.1340%20https://onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1340%20https://onlinelibrary.wiley.com/doi/10.1002/wcms.1340.

[38] Rodolfo Carobene et al. *Github repository: https://github.com/qismib/MEQuVA*. 2021. URL: https://github.com/qismib/MEQuVA.

[39] Stefano Barison, Davide Emilio Galli e Mario Motta. ≪Quantum simulations of molecular systems with intrinsic atomic orbitals≫. In: (2020). arXiv: 2011.08137. URL: http://arxiv.org/abs/2011.08137.

[40] Dmitry A. Fedorov et al. ≪VQE Method: A Short Survey and Recent Developments≫. In: (2021), pp. 1–20. arXiv: 2103.08505. URL: http://arxiv.org/abs/2103.08505.

[41] Edward Farhi, Jeffrey Goldstone e Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. Rapp. tecn. 2014. arXiv: 1411.4028v1.

[42] Harper R. Grimsley et al. ≪An adaptive variational algorithm for exact molecular simulations on a quantum computer≫. In: *Nature Communications* 10.1 (dic. 2019), pp. 1–9. ISSN: 20411723. DOI: 10.1038/s41467-019-10988-2. arXiv: 1812.11173. URL: https://doi.org/10.1038/s41467-019-10988-2.

[43] Jakob S Kottmann et al. *Reducing Qubit Requirements while Maintaining Numerical Precision for the Variational Quantum Eigensolver: A Basis-Set-Free Approach*. Rapp. tecn. 2020. arXiv: 2008.02819v3.

[44] Frank Arute et al. ≪Quantum supremacy using a programmable superconducting processor≫. In: *Nature* 574.7779 (ott. 2019), pp. 505–510. ISSN: 14764687. DOI: 10.1038/s41586-019-1666-5. URL: https://doi.org/10.1038/s41586-019-1666-5.

[45] *On "Quantum Supremacy" — IBM Research Blog*. URL: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/.

[46] Jhonathan Romero Fontalvo. *Variational quantum information processing*. Rapp. tecn. Mag. 2019. URL: https://dash.harvard.edu/handle/1/42029810.

[47] Mario Motta et al. ≪Quantum simulation of electronic structure with a transcorrelated Hamiltonian: Improved accuracy with a smaller footprint on the quantum computer≫. In: *Physical Chemistry Chemical Physics* 22.42 (2020), pp. 24270–24281. ISSN: 14639076. DOI: 10.1039/d0cp04106h. arXiv: arXiv:2006.02488v1.

[48] Jonathan Romero et al. ≪Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz≫. In: *Quantum Science and Technology* 4.1 (2019), pp. 1–18. ISSN: 20589565. DOI: 10.1088/2058-9565/aad3e4. arXiv: 1701.02691.