Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

Quantum software per l'algebra lineare: L'algoritmo HHL e l'IBM quantum experience

Relatore:
Dott. Davide Vodola

Correlatore:
Prof. Elisa Ercolessi

Presentata da:
Pietro Zamberlan

Anno Accademico 2018/2019

Sommario

Nei prossimi anni computer quantistici da 50-100 qubit saranno in grado di eseguire compiti che superano le capacità dei supercomputer classici di oggi, ma fenomeni di rumore nelle porte logiche quantistiche limiteranno la grandezza dei circuiti che si potranno eseguire in maniera affidabile. Attraverso questo tipo di tecnologia sarà possibile svolgere algoritmi nuovi, o già classicamente noti, in maniera più efficiente rispetto ai computer odierni. Ne sono un esempio l'algoritmo di Shor per la fattorizzazione in numeri primi o l'algoritmo di Grover per la ricerca in un database non ordinato. In questa tesi si discute dell'algoritmo HHL (dai propositori: Harrow, Hassidim e Lloyd) per la risoluzione di un sistema lineare, studiando l'algoritmo completo, e le subroutine che lo compongono, sia su simulatori classici che su veri processori quantistici messi a disposizione da IBM Quantum Experience. Se ne ricava che per il caso di una matrice 2×2, opportunamente scelta, l'algoritmo restituisce la corretta soluzione con un alto grado di precisione sui simulatori classici (raggiungendo una fidelity del 99%) ma con una più bassa accuratezza sui qubit reali (fidelity del 84%).

Indice

Introduzione

I computer quantistici di questa e delle prossime generazioni, detti Noisy Intermediate-Scale Quantum computers (NISQ) [1], composti da un numero di qubit nell'ordine di diverse decine [2, 3], anche se ancora affetti da medio/alti livelli di rumore quantistico, in linea di principio potrebbero essere usati per svolgere funzioni che impiegano tempi molto più lunghi a computer classici, oppure di simulare sistemi quantistici a molti corpi.

Sul piano teorico delle scienze della computazione sono già stati sviluppati diversi algoritmi che permettono di risolvere alcuni problemi noti. L'algoritmo di Shor per la fattorizzazione in numeri primi [4] permette uno *speedup* nel trovare la soluzione che è esponenzialmente più veloce della migliore alternativa classica. LK Grover, invece, ha proposto, nel 1996 un algoritmo di ricerca che presenta un miglioramento quadratico nella velocità di risoluzione rispetto all'approccio classico [5]. Computer quantistici di modeste dimensioni potrebbero essere anche utilizzati per la simulazione di sistemi quantistici a molti corpi, un ambito inattaccabile dai computer classici, attraverso la cosiddetta Hamiltonian simulation. Questo approccio risolutivo a problemi computazionali trova anche una diverse applicazioni in algoritmi di Machine Learning, ormai presenti e utilizzati in diversi ambiti di ricerca fisica e non solo. La tecnologia NISQ rappresenta un punto di partenza per lo sviluppo di dispositivi che implementano efficaci sistemi di correzione degli errori.

Un ulteriore problema, ubiquo in fisica è il Linear System Problem o LSP. Nel 2009 A.W. Harrow, A. Hassidim e S. Lloyd hanno presentato un algoritmo quantistico, chiamato HHL, per la soluzione di un sistema lineare [6]. Il LSP, oltre ad essere presente in ogni ambito di ricerca scientifica, è anche rappresentativo di una classe di problemi, detti problemi BQP-completi, a cui appartengono tutti i problemi risolvibili in maniera efficiente su un computer quantistico. Ciò significa che ogni problema che ci si aspetta sarà un giorno risolvibile da un computer quantistico, può essere rielaborato in termini di un sistema lineare [1]. Questo algoritmo fornisce uno *speedup* esponenziale rispetto al migliore algoritmo classico che svolge lo stesso compito, ci si attende quindi che, una volta sviluppata la tecnologia quantistica necessaria, l'HHL avrà un profondo impatto applicativo.

In questa tesi si è voluto testare questo algoritmo attraverso un software kit open source per la computazione quantistica realizzato da IBM. Questo software è chiama-

to Qiskit, e permette di costruire circuiti quantistici, introducendo qubit e applicando porte logiche, e poi di testarli. IBM, nel suo progetto "Quantum Experience" (quantumexperience.ng.bluemix.net/qx/devices [2]) fornisce, oltre a dei simulatori classici locali, anche dei processori quantistici accessibili in remoto, su cui eseguire i circuiti costruiti con Qiskit. Si è trovato che l'algoritmo, per il caso di una matrice 2×2, funziona con una discreta precisione quando simulato in assenza di rumore. Se eseguito su un vero dispositivo quantistico, invece, il circuito produce una soluzione con un grado accettabile di affidabilità, ma che ci si aspetta non sarebbe più tale per matrici di dimensioni più elevate.

Il presente lavoro è organizato come segue:

Nel capitolo 1 si introduce il concetto di computazione quantistica e i suoi elementi fondamentali, come i qubit e gli operatori logici unitari. Si discutono inoltre alcuni teoremi, tra i quali quello di "universalità dei gate unitari" (sez. 1.4) che assicura di poter generare qualunque porta logica (anche tipicamente classica) a partire da un set molto limitato di operatori.

Nel capitolo 2 si presentano i concetti di complessità computazionale, fondamentale nelle scienze dell'informazione, e alcuni algoritmi quantistici di base. Tra questi si analizzano in dettaglio la trasformata di Fourier quantistica e la quantum phase estimation, entrambi utilizzati come subroutine dell'algoritmo HHL, trattato nel capitolo 3.

Nell'ultimo capitolo ci si occupa della definizione quantistica del Linear System Problem e dei particolari, assieme alle limitazioni, dell'algoritmo HHL. Inoltre, nelle ultime sezioni, si presenta la specifica versione del circuito che è stata realizzata, e i risultati dei test condotti su simulatori classici e processori quantistici reali, sia per le componenti costitutive, che per il circuito completo.

Capitolo 1

Cenni di computazione quantistica

1.1 La computazione quantistica

Introdotta come nuovo paradigma di computazione negli anni '80, la computazione quantistica studia i processi di elaborazione di informazione, attraverso sistemi quantistici. Ciò richiede un notevole controllo sperimentale su sistemi quantistici singoli, che svolgono la funzione equivalente ai *bit* nella computazione classica. Tali sistemi sono chiamati *qubit* (quantum-bit) e rappresentano la componente fondamentale di un cosiddetto computer quantistico.

Ci sono diversi motivi di interesse per lo sviluppo su grande scala di tecnologia di questo tipo, il primo, individuato da Richard P. Feynman [7], è la enorme mole di potere computazionale richiesta per la simulazione di processi quantistici, che fa pensare di poter utilizzare calcolatori basati direttamente sulle leggi probabilistiche della meccanica quantistica, così difficili da simulare efficientemente, per ottenere previsioni utili in ambienti scientifici.
Nell'hardware utilizzato in computazione classica, poi, i fenomeni quantistici cominciano a farsi sentire che lo si voglia o meno, a causa delle dimensione sempre più ridotta dei transistor implementati nei circuiti integrati. Considerare allora un computer che tenga conto della natura quantistica della natura si rende forzatamente necessario.

Le leggi della meccanica quantistica poi, introducono una logica che non è quella binaria su cui si basa la computazione classica. Questo tipo di logica, e gli algoritmi su di essa basati, offrono una rapidità nella risoluzione di determinati problemi, che è di molto superiore ai loro corrispettivi classici. Si pensa che nessuna previsione concepibile sul miglioramento del potere computazionale classico possa tenere testa alle possibilità offerte dal futuro della computazione quantistica [8].

1.2 Qubits

La computazione classica si basa sui *bit*, interpretabili a livello teorico come sistemi classici a due stati, caratterizzati da una certa variabile fisica, che supera o meno un definito valore di soglia, e che pone il *bit* nello stato 1, rispetto allo stato 0. Questi sono gli unici due stati possibili.

In computazione quantistica, invece si utilizzano sistemi quantistici a due stati per la trasmissione e l'elaborazione di informazione per la costruzione di circuiti e gate logici. Lo stato di un sistema quantistico si indica con un vettore $|\psi\rangle$ (utilizzando la notazione di Dirac) appartenente a uno spazio vettoriale complesso bidimensionale $\mathbb{C}^2$. A differenza del caso classico, quindi, il sistema può assumere un numero infinito di stati, dati dalla combinazione lineare dei vettori necessari per formare una base ortonormale di autostati, in questo caso solo due. Scelti $|0\rangle$ e $|1\rangle$ come stati computazionali di base lo stato di un qubit si può esprimere come:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \alpha \text{ e } \beta \in \mathbb{C}^2$$

dove l'interpretazione probabilistica del modulo quadro del vettore complesso, dice che misurando $|\psi\rangle$ si ottiene $|0\rangle$ con probabilità $|\alpha|^2$ e $|1\rangle$ con probabilità $|\beta|^2$.

Esempi di sistemi a due stati possono essere la polarizzazione di un fotone, l'allineamento dello spin in un campo magnetico, o gli stati di un elettrone in un atomo. Parlare di sistema quantistico a due stati o di qubit è quindi equivalente, ma in questo elaborato ci si riferirà al modello teorico ideale, utilizzato in computazione quantistica.

Lo stato di un singolo qubit, è quindi rappresentato dai quattro parametri reali che definiscono α e β, la condizione di normalizzazione della probabilità $|\alpha|^2+|\beta|^2 = 1$ riduce questo numero a tre, e ψ può essere rappresentato come:

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\,|1\rangle\right).$$

Si dimostra poi che un fattore di fase non produce effetti osservabili, quindi bastano θ e φ per caratterizzare fisicamente ψ.

Per visualizzare l'evoluzione di un sistema a qubit singolo, è comodo introdurre la cosiddetta *Sfera di Bloch*, una sfera di raggio uno, i cui punti della superficie sono anch'essi identificati da due parametri, intesi come angoli.
Ponendo il centro della sfera nell'origine degli assi di $\mathbb{R}^3$ e associando agli stati $|0\rangle$ e $|1\rangle$, il verso positivo e negativo dell'asse z, come in figura 1.1, si ottiene un comodo strumento visivo per seguire l'evoluzione dello stato.

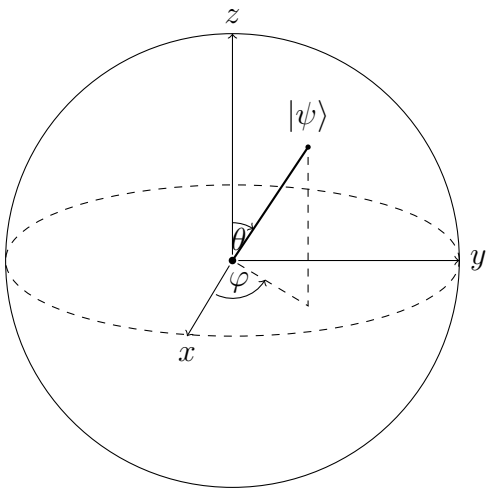


Figura 1.1: Sfera di Bloch

Un computer quantistico, ovviamente, è composto da più di un qubit, o sistemi a due stati, ognuno con due stati di base computazionale $|0\rangle$ e $|1\rangle$.
Complessivamente gli stati di base computazionale per uno spazio di Hilbert n dimensionale sono dati dal prodotto tensoriale di n spazi di Hilbert $\mathbb{C}^2$ per il singolo qubit.

$$|00\ldots00\rangle, |00\ldots01\rangle \ldots |11\ldots11\rangle$$

Anche in questo caso uno stato del sistema complessivo è identificato dalla combinazione lineare dei 2^n vettori di base:

$$|\psi\rangle = \sum_{q_1,\ldots q_n=0,1} \alpha_{q_1,\ldots q_n} |q_1,\ldots q_n\rangle = \sum_{q_{(2)}=0}^{2^n-1} \alpha_q |q_{(2)}\rangle$$

Dove con $q_{(2)}$ si indica la rappresentazione in base binaria di q.
La condizione di normalizzazione della probabilità è espressa in generale da $\sum_{q=0}^{2^n-1} |\alpha_q|^2 = 1$.

Tra gli infiniti stati che un qubit può assumere ce ne sono due, oltre agli autostati della base computazionale, che sono ricorrenti in teoria della computazione quantistica. Questi stati sono indicati dalla notazione $|+\rangle$ e $|-\rangle$:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Si introduce poi una notazione vettoriale per indicare gli stati dei qubit, per cui ad ogni stato $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ è associato il vettore: $\begin{bmatrix} \alpha \\ \beta, \end{bmatrix}$ dove si utilizza la convenzione

di avere il primo coefficiente del vettore che indica l'ampiezza dello stato $|0\rangle$. Lo stato $|+\rangle$ per esempio verrebbe indicato con $|+\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$ Questa notazione può anche essere estesa a sistemi di n-qubit, gli stati di base di un sistema a due qubit sono per esempio:

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

1.3 Gate quantistici

Come per i calcolatori classici, costituiti da fili che trasportano l'informazione (lo stato) dei bit, e porte logiche, che cambiano lo stato dei bit, anche un sistema quantistico che si voglia utilizzare come calcolatore è composto da fili, che possono indicare il trasporto del qubit da un gate all'altro, oppure il trascorrere del tempo, e da porte.

Le porte, o *gate* possono coinvolgere singoli qubit o sistemi multipli. Nel caso di qubit singoli, il corrispettivo classico non banale è solamente la porta NOT, mentre quantisticamente esistono infinite porte logiche che possono cambiare lo stato del qubit. Ciò che agisce su uno stato quantico e ne cambia le caratteristiche è fisicamente un *operatore*, la cui azione su uno stato qualunque è univocamente definita dalla sua azione sugli stati di base.

A ogni operatore agente su un sistema fisico è associata una matrice unitaria che agisce sul vettore nello spazio di Hilbert corrispondente allo stato del sistema.
Indicando con U sia l'operatore che la matrice si ha che, per un singolo qubit:

$$|\psi'\rangle = \mathrm{U}\,|\psi\rangle = \mathrm{U}\,(\alpha\,|0\rangle + \beta\,|1\rangle) \equiv \mathrm{U}\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \begin{bmatrix}u_{00} & u_{01}\\u_{10} & u_{11}\end{bmatrix}\begin{bmatrix}\alpha\\\beta\end{bmatrix} = \begin{bmatrix}\alpha'\\\beta'\end{bmatrix}$$

L'unico vincolo posto sulle matrici che rappresentano univocamente gli operatori è quello dell'unitarietà, che deve valere perché la condizione di normalizzazione rimanga anche per lo stato risultante $|\psi'\rangle = \alpha'\,|0\rangle + \beta'\,|1\rangle$ dopo l'azione della porta logica:

$$\mathrm{UU}^\dagger = I \text{ ovvero } \mathrm{U}^{-1} = \mathrm{U}^\dagger \implies |\alpha'|^2 + |\beta'|^2 = 1.$$

Un qualunque operatore U agente su un qubit è allora interpretabile come rotazione del vettore $|\psi\rangle$ attorno all'origine della sfera di Bloch. Per operatori agenti su sistemi a qubit multipli vale comunque la condizione di unitarietà, ma non esiste uno strumento di visualizzazione immediato quanto la sfera di Bloch.

L'esistenza dell'operatore inverso, qualunque sia l'operatore applicato nel circuito, ha poi un altra implicazione teorica: le porte logiche quantistiche, a differenza di quelle classiche, sono sempre reversibili. Mentre dato l'output di una porta classica, ad esempio

un NAND, non è possible determinare lo stato di input, avendo lo stato finale di uno o più qubit dopo l'azione di un operatore unitario U è possibile riportarli allo stato iniziale, semplicemente applicando U^{-1}. Questo fenomeno, nei computer classici, porta ad un intrinseca perdita di informazione ogniqualvolta un bit passa in un gate, nei calcolatori quantistici questa informazione viene invece conservata fino alla misura del qubit.

Per schematizzare la struttura di un circuito, con i suoi registri di qubit in input e in output, su cui agiscono i gate, si utilizza la notazione riportata in figura 1.2. Per indicare l'azione di un gate si utilizza un riquadro contenente il simbolo del gate, sopra il filo del qubit su cui agisce. Per convenzione i qubit in input si intendono posti nello stato $|0\rangle$ se non specificato direttamente.

Per indicare la misura di un qubit in una determinata base si usa il simbolo riquadrato sulla destra, che indica proprio uno strumento di misura, un filo lasciato vuoto quando arriva al margine destro del disegno è di solito inteso come misurato nella base computazionale, sempre se non specificato diversamente. Dopo una misura l'informazione raccolta dal qubit è passata a un bit classico, indicato con un filo doppio.

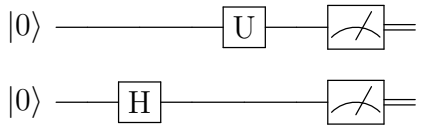


Figura 1.2: Circuito di esempio con registro di 2 qubit, inizialmente nello stato $|0\rangle$, che vengono misurati dopo essere stati sottoposto all'azione dei generici operatori U e H.

1.3.1 Gate notevoli e gate multiqubit

Le porte logiche quantistiche possono quindi essere infinite, in questa sezione si presentano le più note, che insieme a quelle presentate in sezione 1.3.2 costituiscono l'insieme di gate fondamentali, dai quali si possono generare tutti i gate possibili, come vedremo in sezione 1.4.

Partendo dai gate agenti su un singolo qubit, definiti sempre in relazione alla loro azione sui vettori di base computazionale, si hanno:

$$\text{NOT o X: X} \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \equiv$$

Questa porta agisce portando lo stato $|0\rangle$ in $|1\rangle$ e viceversa, è la cosa più simile a una porta NOT classica.

$$\text{Porta Z: Z} \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \equiv$$

Questo gate invece lascia lo stato $|0\rangle$ invariato, e porta $|1\rangle$ in $-|1\rangle$.

$$\text{Porta Y: Y} \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \equiv -\boxed{\text{Y}}-$$

L'azione di questa porta non è immediata ($|0\rangle \to -i\,|1\rangle\,, |1\rangle \to i\,|1\rangle$) ma è utile riportare assieme queste tre queste porte se si nota che non sono altro che σ_x, σ_y e σ_z ovvero le matrici di Pauli.

Un'altra porta fondamentale è la cosiddetta porta Hadamard (*Hadamard gate*):

$$\text{H} \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \equiv -\boxed{H}-$$

che porta $|0\rangle$ nello stato $|+\rangle$ e $|1\rangle$ nello stato $|-\rangle$.

Le matrici di Pauli $\text{X}, \text{Y}, \text{Z}$ permettono, come è noto, di definire degli ulteriori operatori, chiamati *rotazioni* lungo x, y e z, quando vengono esponenziate:

$$\text{R}_\text{x}(\theta) \equiv e^{-i\theta X/2} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2}, \end{bmatrix} \tag{1.1}$$

$$\text{R}_\text{y}(\theta) \equiv e^{-i\theta Y/2} = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \tag{1.2}$$

$$\text{R}_\text{z}(\theta) \equiv e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}. \tag{1.3}$$

L'effetto di queste porte è proprio quello di ruotare di un angolo θ il vettore di Bloch attorno all'asse a cui si riferiscono e risultano particolarmente importanti perché con esse è possibile generare una qualunque operazione unitaria su singolo qubit.

I gate che agiscono su un sistema a n-qubit sono identificati da una matrice $N \times N$ con $N = 2^n$. Ad esempio l'azione dell'operatore U definito da

$$\text{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

su uno stato iniziale

$$|10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

produce lo stato

$$\text{U}\,|10\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle\,.$$

1.3.2 Controlled operations

La matrice U delle esempio 1.3.1 non è stata scelta a caso, questo operatore rappresenta infatti il più importante e più utilizzato operatore a multiqubit nei circuiti quantistici e prende il nome di CNOT gate dove la C nel nome sta per *controlled*. Esiste infatti una classe di operazioni che si possono svolgere su più di un qubit in cui uno tra questi svolge l'operazione di *controllare* l'effettiva applicazione della matrice sui rimanenti qubit in considerazione. Se il *control qubit* è in uno stato determinato allora la porta viene applicata sui *target qubit*, in caso contrario, questi ultimi non vengono toccati. Nel caso del CNOT precedente, se il control qubit si trova nello stato $|1\rangle$, sul target qubit viene applicata la porta NOT = X altrimenti se il control qubit si trova in $|0\rangle$ non cambia nulla. È importante notare che sul control qubit non viene mai eseguita alcuna operazione e il suo stato rimane comunque invariato.

Qualunque operatore U agisca su uno o più qubit può essere trasformato una *controlled operation* aggiungendo un qubit di controllo. Ci si riferisce in genere a questo tipo di operatori con il nome di *controlled* − U o $^{(c)}$U, dove al simbolo U va sostituito il nome del gate originale.

Il concetto di controlled operation può essere esteso anche a molti qubit, in cui più di uno svolgono la funzione di controllo e devono essere tutti in un determinato stato perché, la porta venga applicata.

Un ulteriore applicazione delle control operations sta nella possibilità di creare degli SWAP gates, la cui funzione è quella di invertire gli stati dei qubit si cui agiscono.

$$\text{SWAP}\,|01\rangle = |10\rangle$$

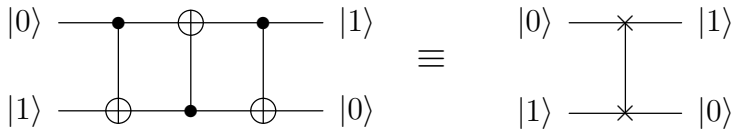


Figura 1.3: SWAP gate

1.4 L'universalità dei gate unitari

Come anticipato in sezione 1.3.1 ogni operatore unitario agente su un singolo qubit può essere generato da una famiglia molto ristretta di operatori di base, le rotazioni (1.1, 1.2, 1.3). Dimostriamo ora [8] che un qualunque operatore U è esprimibile nella sua forma matriciale come:

$$\text{U} = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta) \tag{1.4}$$

in cui $e^{i\alpha}$ è un fattore di fase.

Convincersi dell'espressione (1.4) è semplice se si considera che una qualunque matrice unitaria 2×2 ha forma parametrica:

$$\begin{bmatrix} a & -b^* \\ b & a^* \end{bmatrix}$$

e un'espressione del genere deriva proprio dal prodotto di $R_z R_y R_z$:

$$\begin{bmatrix} e^{-i\frac{\beta}{2}} & 0 \\ 0 & e^{i\frac{\beta}{2}} \end{bmatrix} \begin{bmatrix} \cos\frac{\gamma}{2} & -\sin\frac{\gamma}{2} \\ \sin\frac{\gamma}{2} & \cos\frac{\gamma}{2} \end{bmatrix} \begin{bmatrix} e^{-i\frac{\delta}{2}} & 0 \\ 0 & e^{i\frac{\delta}{2}} \end{bmatrix} = \begin{bmatrix} e^{i(-\beta/2-\delta/2)\cos\frac{\gamma}{2}} & -e^{i(-\beta/2+\delta/2)\sin\frac{\gamma}{2}} \\ e^{i(+\beta/2-\delta/2)\sin\frac{\gamma}{2}} & e^{i(+\beta/2+\delta/2)\cos\frac{\gamma}{2}}, \end{bmatrix}$$

alla quale va ad aggiungersi il fattore $e^{i\alpha}$.

Questo è un risultato fondamentale, perché permette di avere a disposizione, almeno sul piano teorico, ogni tipo di gate si voglia realizzare, fig. 1.4.

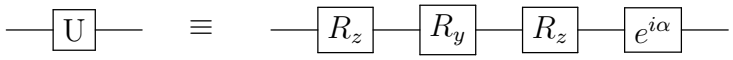


Figura 1.4: Il gate U è equivalente al circuito che implementa separatamente A, B e C

Con queste rotazioni e con la porta C − NOT si è poi in grado [8] di generare anche una qualunque controlled operation seguendo lo schema di figura 1.5.

Si verifica infatti che definendo delle matrici A, B e C come:

$$\mathrm{A} \equiv R_z(\beta)R_y(\frac{\gamma}{2}),\ \mathrm{B} \equiv R_y(-\frac{\gamma}{2})R_z(-\frac{\delta+\beta}{2}),\ \mathrm{C} \equiv R_z(\frac{\delta-\beta}{2}),$$

per cui valga $ABC = I$, è possibile riformulare l'operatore U come:

$$\mathrm{U} = e^{i\alpha} AXBXC. \tag{1.5}$$

Le porte NOT vengono quindi applicate solo se il control qubit è nello stato voluto, e in caso contrario l'azione dei gate A, B e C non varia lo stato del target qubit.

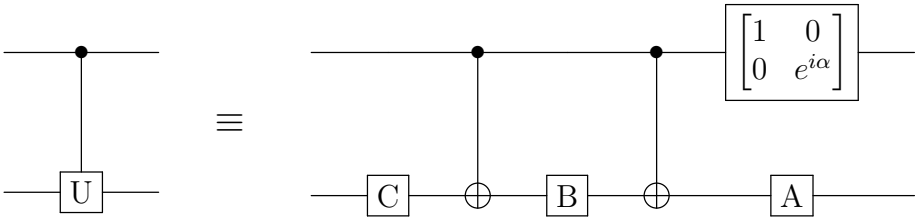


Figura 1.5: Implementazione di una generica operazione ${}^{(c)}\mathrm{U}$

Il fattore di fase della formula 1.5, che dovrebbe essere un operazione controllata, si riduce all'azione del gate:

$$e^{i\alpha} I = \begin{bmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{bmatrix} \tag{1.6}$$

sul control qubit, se si nota che

$${}^{(c)}\begin{bmatrix} e^{i\alpha} & 0 \\ 0 & e^{i\alpha} \end{bmatrix} : \ |00\rangle \to |00\rangle \,, \ |01\rangle \to |01\rangle \,, \ |10\rangle \to e^{i\alpha} |10\rangle \,, \ |11\rangle \to e^{i\alpha} |11\rangle$$

è l'azione complessiva del gate di fase controllato. È quindi perfettamente giustificabile la sua applicazione direttamente sul control qubit.

Abbiamo dimostrato che qualunque operatore agente su uno spazio di Hilbert unidimensionale può essere espresso come prodotto di operatori noti, agenti su singolo qubit. Si può dimostrare anche che utilizzando dei gate CNOT, Hadamard, e dei gate di fase simili a quello nell'espressione 1.6, si può generare, con approssimazione arbitraria, l'azione di un gate unitario agente su uno spazio di Hilbert 2^d dimensionale, ovvero un sistema di d qubit [9].

1.5 Teorema della non clonazione

Un teorema interessante in teoria della computazione quantistica è il cosiddetto teorema dalla non clonazione. Questo teorema dichiara un risultato a prima vista alquanto sorprendente, ovvero che non è possibile creare, per quanto complesso il circuito quantistico a disposizione, una copia esatta dello stato di un qubit (e quindi di più qubit), senza modificare l'originale.

Questo risultato si dimostra ammettendo che esista un circuito di clonazione la cui azione sia equivalente all'operatore U, agente su due registri. Il primo registro contiene uno stato quantistico puro, ovvero autostato della base su cui si svolgerà la misura, chiamato $|\psi\rangle$, il secondo contiene invece uno stato generico $|s\rangle$. Lo stato iniziale risulta quindi

$$|\psi\rangle \otimes |s\rangle \,.$$

Ammettendo che questo circuito possa clonare lo stato $|\psi\rangle$ sul secondo registro, sovrascrivendo $|s\rangle$ ipotizziamo di farlo agire due volte, cambiando lo stato iniziale puro nel primo registro.

$$\begin{aligned} \mathrm{U}(|\psi\rangle \otimes |s\rangle) &= |\psi\rangle \otimes |\psi\rangle \\ \mathrm{U}(|\phi\rangle \otimes |s\rangle) &= |\phi\rangle \otimes |\phi\rangle \,. \end{aligned}$$

Facendo il prodotto BraKet delle precedenti espressioni si ottiene l'equazione:

$$\langle\psi|\phi\rangle = (\langle\psi|\phi\rangle)^2,$$

la quale implica che gli stati $|\psi\rangle$ e $|\phi\rangle$ sono uguali oppure ortogonali. Anche assumendo l'esistenza di un programma quantistico generale per la clonazione di uno stato qualsiasi, si incappa quindi in dei limiti di applicazione molto stringenti.

Capitolo 2

Algoritmi quantistici di base e la loro utilità

Nel presente capitolo vengono presentati alcuni concetti fondamentali della teoria computazionale quantistica, in particolare quelli più utili all'elaborazione teorica di un algoritmo, da eseguire quindi su un simulatore. Non viene presentato il concetto di rumore quantistico, effetto molto presente nell'esecuzione di programmi su processori quantistici reali.

Dal punto di vista ideale restano importanti concetti come la complessità computazionale, che permette di valutare l'efficienza di un algoritmo, e proprietà come il parallelismo quantistico, entrambi presentati nelle successive sezioni. Sono poi spiegati in relativo dettaglio gli algoritmi di trasformata di Fourier quantistica e di *phase estimation*, che sono routine fondamentali utilizzate nella gran parte degli algoritmi quantistici più complessi, compreso l'algoritmo HHL per l'inversione di matrice, a cui si dedica il capitolo 3.

2.1 La complessità computazionale

Complessità computazionale è il termine utilizzato per indicare i requisiti di *tempo* e *spazio* richiesti da un algoritmo per essere eseguito su una macchina di Turing[1]. In questa trattazione, è utile riferirsi solo ai requisiti temporali di un algoritmo, così da poter trattare gli algoritmi quantistici di interesse al netto delle richieste materiali del sistema fisico in cui è implementato.

La classe di problemi più semplice è quella dei problemi *decisionali*, ovvero la cui risposta può essere solamente un *sì* o un *no*. Un problema di questo tipo può essere quello di determinare se un numero, all'interno di una lista, sia primo oppure no. Questa classe di problemi è formulabile in termini di un cosiddetto problema di linguaggio: in maniera

[1]Possiamo chiamare macchina di Turing uno strumento astratto di computazione in grado di eseguire algoritmi il cui risultato viene scritto su una memoria illimitata.

del tutto generale prendiamo Σ come un insieme finito di simboli, che chiamiamo *alfabeto*, e Σ^* l'insieme di tutte le possibili combinazioni finite di questi simboli. Considerando poi il cosiddetto *linguaggio* L sottoinsieme di Σ^*, il problema decisionale più generale è quello di determinare se una stringa di caratteri x sia o meno parte del linguaggio L secondo l'alfabeto Σ.
Nell'esempio precedente il linguaggio è costituito dall'insieme dei numeri primi, espressi in una base i cui simboli appartengono all'alfabeto.

Un problema è detto TIME($f(n)$), con n che quantifica la dimensione dell'input, se esiste una macchina di Turing che può essere programmata per risolverlo in un tempo che va come $\sim O(f(n))$ [2]. Si dice allora che un problema è risolvibile in tempo *polinomiale* se $t_{sol} \propto O(n^k)$ per un certo k finito. In un problema di linguaggio, per esempio, n corrisponde alla lunghezza di x. L'insieme dei linguaggi risolvibili in un tempo polinomiale prende il nome di classe di complessità **P**. Esistono molte classi di complessità, e la teoria della complessità computazionale si occupa di analizzare le relazioni tra queste classi [8].

La classe **NP** (*non deterministic polynomial time*), per esempio è definibile come l'insieme di linguaggi o problemi la cui soluzione, se trovata, è verificabile in un tempo polinomiale. Ciò produce la relazione $\mathbf{P} \subseteq \mathbf{NP}$, poiché ogni problema in **P**, una volta trovata la soluzione, è verificabile / risolvibile in un tempo polinomiale. Una questione aperta nelle scienze informatiche è invece quella di provare una congettura più che condivisa, ovvero che $\mathbf{P} \neq \mathbf{NP}$.

Nella teoria della computazione quantistica si punta a trovare algoritmi il cui andamento del tempo impiegato a trovare la soluzione sia di ordine inferiore rispetto ai loro corrispondenti classici. Passare da un algoritmo classico in TIME(e^n) a uno quantistico in TIME(n^k) produce un miglioramento esponenziale. Il passaggio da un tempo risolutivo più che polinomiale a uno *sub* polinomiale rappresenta, per un problema, il superamento del confine tra *non efficienza* ed *efficienza* dell'algoritmo che lo risolve, un algoritmo è infatti definito *efficiente* se e solo se è in TIME(n^k)

2.2 Algoritmi quantistici

Per studiare l'algoritmo HHL per l'inversione di matrice e le subroutines in esso utilizzate, è bene introdurre il concetto di algoritmo quantistico e identificare quando questo sia conveniente rispetto a un algoritmo classico.

2.2.1 Parallelismo quantistico

Una caratteristica fondamentale dei sistemi quantistici, che viene spesso sfruttata negli algoritmi di questo tipo, è le possibilità di trovare un qubit, che potrebbe rappresentare

[2] Si utilizza la notazione $O-grande$ di $f(x)$ per indicare una funzione il cui valore, se moltiplicato per un opportuna costante finita c, maggiora il valore di f per tutti gli x maggiori di un certo x_0

l'output di una funzione $f(x)$, in una sovrapposizione di stati, in cui ognuno rappresenta il valore di f valutato per diversi valori di x. In poche parole, un algoritmo quantistico può valutare una funzione per diversi valori dell'input contemporaneamente.

Per dimostrare questa proprietà, chiamata *parallelismo quantistico* supponiamo di avere una semplice funzione con dominio e codominio di un bit $f(x) : \{0,1\} \to \{0,1\}$ e un computer quantistico a 2 qubit che agisce sullo stato iniziale $|x,y\rangle$ generando lo stato $|x, y \oplus f(x)\rangle$, dove $f(x) \in \{0,1\}$ e x, y sono stati arbitrari. Chiamando U_f la trasformazione effettuata dal circuito si vede che se $y = 0$ lo stato nel secondo registro, detto *target register* è semplicemente il valore di $f(x)$.
Se lo stato x non appartiene ala base computazionale, ma si ha $|x\rangle = |+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ l'azione del circuito è:

$$U_f\,|x,y\rangle = U_f\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|y\rangle = \frac{1}{\sqrt{2}}\left(U_f\,|0,y\rangle + U_f\,|1,y\rangle\right) = \frac{|0,f(0)\rangle + |1,f(1)\rangle}{\sqrt{2}}. \quad (2.1)$$

Questo è uno stato molto interessante e rappresenta l'espressione fondamentale del parallelismo quantistico: lo stato finale del secondo registro è una sovrapposizione di vettori di stato identificati dai due valori assunti dalla funzione $f(x)$.
Generalizzare questo circuito a funzioni su un numero arbitrario di bit risulta piuttosto semplice operando la cosiddetta *trasformata di Hadamard*, indicata dal simbolo $H^{\otimes n}$ sul numero n di qubits del primo registro. Questo operatore, ovviamente unitario, porta tutti i qubit nello stato $|+\rangle$, lasciando il registro nello stato:

$$\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)+\cdots = \frac{|00\ldots0\rangle + |00\ldots1\rangle + \cdots + |11\ldots1\rangle}{\sqrt{2^n}} = \frac{\sum_x |x\rangle}{\sqrt{2^n}}. \quad (2.2)$$

Applicare ora l'operatore U_f allo stato $\frac{\sum_x |x\rangle}{\sqrt{2^n}}\,|0\rangle$ genera lo stato

$$\frac{1}{\sqrt{2^n}}\sum_x |x\rangle\,|f(x)\rangle\,,$$

in cui, di nuovo, il sistema quantistico si trova in una sovrapposizione di stati i cui vettori sono identificati contemporaneamente dai valori di $f(x)$ per ogni x del dominio della funzione.

I valori valutati dall'operatore U, però, non sono tanto utili quanto sembra. Una misura dello stato finale restituisce solo uno degli stati $\frac{|x_0\rangle}{\sqrt{2}}\,|f(x_0)\rangle$ e l'informazione relativa agli altri valori è irrimediabilmente persa, ciò vanifica in parte i vantaggi di questo approccio. Il fenomeno del parallelismo quantistico, pur non permettendo di calcolare qualunque funzione con una sola operazione di un circuito, non è solo una curiosità emergente dalla meccanica quantistica, ha infatti delle applicazioni pratiche di cui si fornisce un esempio nella prossima sezione, dedicata all'algoritmo di Deutsch-Jozsa.

2.2.2 Un esempio: algoritmo Deutsch-Jozsa

L'algoritmo di Deutsch-Jozsa permette di risolvere il *problema di Deutsch* che può essere formulato come segue: sia data una funzione $f : [0, 2^n - 1] \to 0, 1$ che può essere solo di due tipi: o costante per ogni valore del dominio, oppure bilanciata, ovvero uguale a 0 per metà dei valori del dominio, e uguale a 1 per l'altra metà. L'obiettivo è ricavare con certezza e con il minor numero possibile di esecuzioni della funzione, la tipologia a cui corrisponde f, che si può assumere essere molto dispendiosa da calcolare in termini di risorse.

Classicamente, sarà necessario calcolare f un numero di volte pari a $\frac{2^n-1}{2} + 1$ nel peggiore dei casi, per avere una risposta certa al problema. Quantisticamente, imponendo in più solo la condizione che f sia calcolata via trasformazioni unitarie, basterà una valutazione di f per avere la soluzione.

Ciò che serve è un circuito quantistico ideale con $n+1$ qubits, diviso in n qubit in $|0\rangle$ per il codiddetto *registro di query*, e un qubit in $|1\rangle$ per salvare il risultato della funzione, detto *registro di risposta*.

L'algoritmo comincia con una trasformata di Hadamard $H^{\otimes n}$ sui primi n qubits, e una semplice porta H applicata all'ultimo qubit, che porta il sistema nello stato:

$$H^{\otimes n} |0\rangle^{\otimes n} H |1\rangle = \sum_{x \in \{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right].$$

Applicando ora il gate unitario $U_f : |x, y\rangle \to |x, y \oplus f(x)\rangle$ ai due registri, si ottiene:

$$U \sum_x \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] = \sum_x \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} \right] =$$
$$= \sum_x \frac{(-1)^{f(x)} |x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$$

Ora il risultato della funzione è racchiuso nell'ampiezza della sovrapposizione dei qubit del primo registro, applicando un ulteriore trasformata di Hadamard sul *query register*, si ottiene [8] lo stato:

$$H^{\otimes n} |x_1, \ldots, x_n\rangle = \frac{\sum_{z_1, \ldots z_n} (-1)^{x_1 z_1 + .. x_n z_n} |z_1, \ldots, z_n\rangle}{\sqrt{2^n}} = \frac{\sum_z (-1)^{x \cdot z} |z\rangle}{\sqrt{2^n}},$$

Per il sistema complessivo, quindi, lo stato finale $|\psi\rangle$ è uguale a:

$$\sum_z \sum_x \frac{(-1)^{x \cdot z + f(x)} |z\rangle}{2^n} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right].$$

Per valutare f basta ora misurare il registro di query: se la funzione assume valore costante, l'ampiezza complessiva per lo stato $\sum_x (-1)^{f(x)}/2^n |0\rangle^{\otimes n}$ vale +1 o -1 a seconda

del valore assunto da $f(x)$, ciò comporta che le ampiezze su tutti le altre combinazioni siano 0 e la misura restituisca tutti i qubit nello stato $|0\rangle$. Se f è invece bilanciata, i contributo positivi e negativi all'ampiezza dello stato $|0\rangle^{\otimes n}$ si elidono, e la misura restituisce almeno un qubit in uno stato diverso da $|0\rangle$.
Si è quindi riusciti a determinare la natura di f attraverso una sola valutazione della stessa.

2.3 Algoritmi di base

Ogni tipo di algoritmo eseguibile su un processore classico può essere replicato da un calcolatore quantistico. La differenza fondamentale sta nel fatto che, come evidenziato nella sezione 1.3, le porte logiche quantistiche sono reversibili, essendo rappresentate da operatori unitari, mentre la maggior parte di quelle classiche, come le porte NAND, NOR, usate per la realizzazione di circuiti integrati, non lo sono. Si può dimostrare [8] che esiste una porta logica unitaria, la porta *Toffoli*, con cui si può replicare l'azione di un NAND gate, e con essa quindi ogni algoritmo in logica classica. La porta Toffoli agisce su 3 qubit: i primi due fanno da controllo sull'ultimo, i cui autostati ($|0\rangle$ e $|1\rangle$) vengono invertiti se e solo se i qubit uno e due sono entrambi nello stato $|1\rangle$, in fig. 2.1 se ne riporta lo schema circuitale.

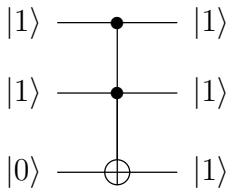


Figura 2.1: Esempio dell'azione di un Toffoli Gate

Appurato che un computer classico può essere simulato da un computer quantistico rimane da discutere quando quest'ultimo rappresenta un alternativa conveniente al primo.

Esistono due classi generali di algoritmi quantistici complessi che soddisfano la richiesta di essere più rapidi a risolvere problemi noti rispetto ad algoritmo classici, essi sono la *Trasformata di Fourier quantistica*, o QFT (sez. 2.3.1) e l'algoritmo di *ricerca quantistica* [8]. Quest'ultimo è importante perché per molti problemi, il miglior algoritmo classico conosciuto è in qualche modo equivalente a una ricerca per tentativi, di cui questo algoritmo fornisce un incremento di velocità quadratico. Ciò permette una adattamento relativamente semplice di questi algoritmi classici al caso quantistico, fornendo un miglioramento di prestazione non impressionante ma comunque utile. Da questi algoritmi di base se ne ricavano un numero non indifferente di altri, tra i quali la *phase estimation* o lo stesso algoritmo HHL, più rapidi rispetto ai corrispettivi classici, ma con le limitazioni di principio imposte dalla meccanica quantistica. Spesso, infatti, pur avendo a

disposizione un algoritmo quantistico più rapido del corrispettivo classico, si incontrano in *input* e in *output* degli importanti rallentamenti, capaci di riportare l'andamento nel tempo della risoluzione del problema allo stato del caso classico.

Questo genere di limitazioni è ciò che rende l'ambito della ricerca di algoritmi quantistici così complesso.

2.3.1 Trasformata di Fourier quantistica

La *trasformata di Fourier* è un processo matematico che permette di scomporre una funzione numerica, periodica o meno, nelle sue componenti di frequenza fondamentali, anche infinite. In fisica le sue applicazioni sono molteplici, e nelle scienze informatiche la sua formulazione in termini quantistici, è utilizzata per esempio nella soluzione del problema del logaritmo discreto o vari problemi di fattorizzazione, oltre che nell'algoritmo HHL, discusso nel capitolo 3.

È bene ricordare la definizione formale della trasformata per poi passare alla costruzione dell'algoritmo.
In ambito discreto, la trasformata di Fourier, o DFT agisce come una matrice quadrata D di dimensione N con componenti $D_{jk} = \frac{1}{\sqrt{N}} e^{i2\pi jk/N}$ su un vettore di dimensione analoga $\vec{x}$. Le colonne della matrice D formano una base ortonormale, chiamata *base di Fourier*, corrispondente ai vettori che risultano dalla DFT applicata ai vettori della base canonica.

In maniera simile, la *trasformata di Fourier quantistica* o QFT è definita dall'azione sui vettori della base computazionale di un sistema N dimensionale come:

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{i2\pi \frac{k}{N} j} |k\rangle \,, \tag{2.3}$$

con $0 \leq j \leq N-1, \in \mathbb{Z}$ e $|k\rangle$ vettore della base computazionale.
L'azione su un generico stato $|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$ risulta, secondo la definizione:

$$\mathrm{QFT}\,|\psi\rangle = \mathrm{QFT} \sum_{j=0}^{N-1} x_j |j\rangle = \sum_{k=0}^{N-1} y_j |k\rangle \tag{2.4}$$

in cui le ampiezze y_j sono la trasformata di Fourier discreta sul vettore delle ampiezze $\vec{x}$.

Per ottenere un'espressione più intuitiva della QFT torniamo all'azione che l'operatore ha sui vettori di base (2.3), nella quale possiamo riscrivere l'intero $k \in [0, N-1]$ in notazione binaria:

$$k = k_1 k_2 \ldots k_n = k_1 2^{n-1} + k_2 2^{n-2} + \cdots + k_{n-1} 2^1 + k_n 2^0 = \sum_{l=1}^{n} k_l 2^{n-l}$$

con $2^n = N$ e $k_l = 1, 0$. Si ha quindi:

$$\begin{aligned}\mathrm{QFT}\,|j\rangle =& \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\exp\left(i\frac{2\pi}{N}\sum k_l 2^{n-l} j\right)|k\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\exp\left(i2\pi\sum k_l 2^{-l} j\right)|k\rangle \\ =& \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{i2\pi j 0.k_1} e^{i2\pi j 0.0k_2}\dots e^{i2\pi j 0.0\dots k_n}\,|k_1\dots k_n\rangle\end{aligned}$$

Ora, ricordando che k_l può assumere solo valore 0 o 1, si può esprimere l'ultima sommatoria come:

$$\begin{aligned}&\frac{1}{\sqrt{N}}\sum_{k=0}^{N-2} e^{i2\pi j 0.k_1} e^{i2\pi j 0.0k_2}\dots e^{i2\pi j 0.0\dots k_{n-1}}\,|k_1\dots k_{n-2}\rangle\left(|0\rangle + e^{i2\pi j 2^{-n}}\,|1\rangle\right) = \\ &\frac{1}{\sqrt{N}}\left(|0\rangle + e^{i2\pi j 2^{-1}}\,|1\rangle\right)\left(|0\rangle + e^{i2\pi j 2^{-2}}\,|1\rangle\right)\dots\left(|0\rangle + e^{i2\pi j 2^{-n}}\,|1\rangle\right).\end{aligned}$$

Scrivendo ora anche $j = j_1 j_2 \dots j_n$ in base binaria, e notando che in questa notazione vale

$$2^{-m} j = j_1 \dots j_{n-m}.j_{n-m+1}\dots j_n$$

si può riscrivere l'ultima espressione come:

$$\frac{1}{\sqrt{N}}\left(|0\rangle + e^{i2\pi 0.j_n}\,|1\rangle\right)\left(|0\rangle + e^{i2\pi 0.j_{n-1}j_n}\,|1\rangle\right)\dots\left(|0\rangle + e^{i2\pi 0.j_1\dots j_n}\,|1\rangle\right) \tag{2.5}$$

perché, in qualunque combinazione, le cifre a sinistra del punto rendono il termine esponenziale uguale a 1, e possono quindi essere trascurate.

La (2.5) è la formulazione più intuitiva della QFT, in cui si può vedere che l'informazione dello stato di base su cui sta agendo, è ora distribuita su N vettori.

L'analisi di un circuito di QFT si basa su questa formula, se ne riporta uno schema in figura 2.2. Si parte da un registro di n-qubit con cui esprimere lo stato di base $|j\rangle = |j_1 j_2 \dots j_n\rangle$ su cui agisce la trasformata, si applica poi, sul primo qubit, inizialmente nello stato $|j_1\rangle$ una porta Hadamard, il cui risultato è:

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi 0.j_1}\,|1\rangle\right) \tag{2.6}$$

In base 2 infatti $0.1_{(2)} = \frac{1}{2}_{(10)}$ quindi

$$\left.\begin{aligned}&\text{se } j_1 = 0 \longrightarrow H\,|j_1\rangle = |+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi 0}\,|1\rangle\right) \\ &\text{se } j_1 = 1 \longrightarrow H\,|j_1\rangle = |-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i2\pi\frac{1}{2}}\,|1\rangle\right)\end{aligned}\right\} = (2.6).$$

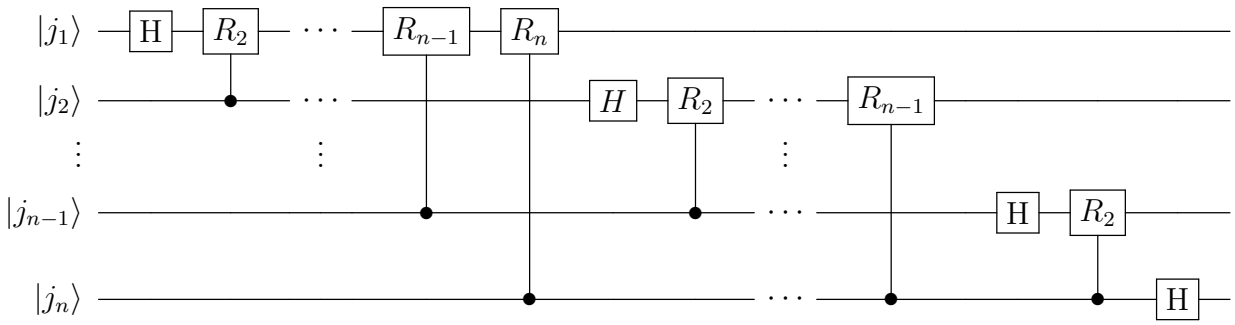


Figura 2.2: Il circuito da applicare a uno stato di base $|j_1 \cdots j_n\rangle$ per ottenerne la trasformata di Fourier, o QFT

Con un ragionamento simile, si può poi studiare l'azione del gate $R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{i2\pi/2^k} \end{bmatrix}$. Implementando R_2 come *controlled operation (c)* condizionato da $|j_2\rangle$, si ottiene:

$$^{(c)}R_2 H\,|j_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi 0.j_1 j_2}\,|1\rangle)$$

In generale l'azione del gate $^{(c)}R_k$ condizionato dall' *l-esimo* qubit aggiunge una fase $\exp\left(i2\pi 2^{-k} j_l\right)$ davanti alla componente dello stato $|1\rangle$. Continuando ad applicare porte $^{(c)}R_k$ condizionate da $|j_k\rangle$ sul primo qubit si ottiene infine lo stato *complessivo*:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{i2\pi 0.j_1 \ldots j_n}\,|1\rangle)\,|j_2 \ldots j_n\rangle$$

Considerando l'*l-esimo* dei rimanenti qubit ($l \in [1, n]$), su di esso si applica una successione di $n - l$ rotazioni controllate $^{(c)}R_k$ ($k \in [n - l + 1]$), come indicato in fig. 2.2, lo stato finale risulta:

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{i2\pi 0.j_1 \ldots j_n}\,|1\rangle\right)\left(|0\rangle + e^{i2\pi 0.j_2 \ldots j_n}\,|1\rangle\right)\ldots\left(|0\rangle + e^{i2\pi 0.j_n}\,|1\rangle\right).$$

Confrontandolo con la (2.5), si riconosce nell'ultima espressione l'azione della trasformata di Fourier sullo stato $|j_n \ldots j_1\rangle$, per ottenere il risultato di QFT $|j_1 \ldots j_n\rangle$ basta allora applicare dei SWAP per invertire l'ordine dei qubit.

2.3.2 Quantum Phase estimation

La QFT è utilizzata all'interno di un algoritmo molto utile, chiamato Quantum Phase Estimation, capace di ottenere, dato un operatore unitario U una approssimazione arbitrariamente precisa dei suoi autovalori λ_i. Il nome deriva dal fatto che se l'operatore è unitario allora è scrivibile come $U = e^{iA}$, e i suoi autovalori sono a loro volta scrivibili

come $\lambda_i = e^{i2\pi\phi_i}$ essendo delle fasi complesse. In questa scrittura $\phi \in [0,1]$ in modo che sia comodo esprimerlo successivamente in base 2.

Supponendo che valga la relazione $U\,|u\rangle = e^{i2\pi\phi}\,|u\rangle$, con $|u\rangle$ autovettore, un'approssimazione a t-bit di $\phi \approx \tilde{\phi} = 0.\phi_1\phi_2\ldots\phi_t$ permette un'approssimazione a t-bit di λ. Supponiamo inoltre di essere in grado di preparare con assoluta precisione l'autostato $|u\rangle$, su cui far agire l'operatore controlled-U^{2^j} la cui azione è:

$$U^{2^0}|u\rangle = U|u\rangle = e^{i2\pi\phi}|u\rangle\,, \quad U^{2^1}|u\rangle = UU|u\rangle = Ue^{i2\pi\phi}\,|u\rangle = e^{i2\pi(2^1\phi)}|u\rangle\,,$$
$$\ldots\ U^{2^j}|u\rangle = e^{i2\pi(2^j\phi)}|u\rangle \quad (2.7)$$

Il circuito per la *phase estimation* si basa su due registri di qubit, il primo, contenente t qubit inizialmente posti in $|0\rangle$, serve a memorizzare l'approssimazione a t-qubit della fase che vogliamo stimare, e per questo, più aumentano i qubit in questo registro, più sarà precisa la nostra stima. Il secondo registro invece contiene i qubit necessari a rappresentare $|u\rangle$. Lo stato iniziale del sistema è allora scrivibile come

$$|0\rangle^{\otimes t}|u\rangle$$

Nella prima parte del circuito, tutti i qubit nel primo registro vengono portati in $|+\rangle$ attraverso una porta Hadamard su ognuno, generando lo stato

$$|+\rangle^{\otimes t}|u\rangle$$

In successione vengono poi applicate, su $|u\rangle$ le porte U^{2^j} controllate dal j-esimo qubit del primo registo, come mostrato in figura 2.3.
Ci si trova ora nello stato

$$\bigotimes_{j=0}^{t-1} \frac{|0\rangle|u\rangle + U^{2^j}\,|1\rangle\,|u\rangle}{\sqrt{2}}$$

in cui si vede che l'operatore U^{2^j} agisce solo su $|1\rangle$ perché controllato dal j-esimo $|+\rangle$.
Il secondo registro può essere considerato come invariato, mentre il primo è riscrivibile come

$$\bigotimes_{j=0}^{t-1} \frac{|0\rangle + e^{i2\pi 2^j\phi}\,|1\rangle}{\sqrt{2}} =$$
$$\left(\frac{|0\rangle + e^{i2\pi 2^{t-1}\phi}\,|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + e^{i2\pi 2^{t-2}\phi}\,|1\rangle}{\sqrt{2}}\right)\ldots\left(\frac{|0\rangle + e^{i2\pi 2^{0}\phi}\,|1\rangle}{\sqrt{2}}\right) =$$
$$\left(\frac{|0\rangle + e^{i2\pi 0.\phi_t}\,|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + e^{i2\pi 0.\phi_{t-1}\phi_t}\,|1\rangle}{\sqrt{2}}\right)\ldots\left(\frac{|0\rangle + e^{i2\pi 0.\phi_1\ldots\phi_t}\,|1\rangle}{\sqrt{2}}\right).$$

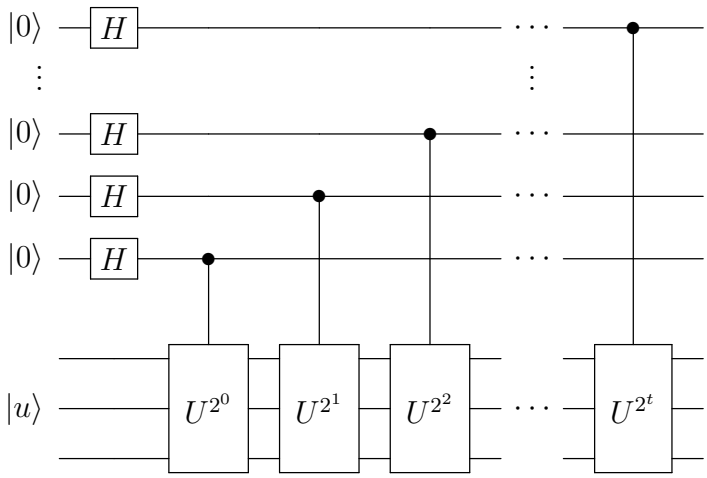


Figura 2.3: Schema del circuito di Quantum Phase Estimation, o PHE.

che se confrontata con la (2.5) si vede essere la condizione finale di un vettore $|\phi_1 \dots \phi_t\rangle$ su cui è stata applicata la trasformata di Fourier. Applicando allora al primo registro la QFT†, ovvero la trasformata inversa e misurandolo sulla base computazionale si ottiene l'approssimazione in t-bit di ϕ.

Ricaviamo ora il numero t di qubit necessari per ottenere una stima di ϕ secondo una precisione da noi scelta, caratterizzata dal numero ξ intero positivo:
Detto b l'intero tale che $\frac{b}{2^t} = 0.b_1 \dots b_t < \phi$ sia l'approssimazione in t-bit più vicina possibile a ϕ e minore di questo, la differenza tra ϕ e b è $\delta \doteq \phi - \frac{b}{2^t} \leq 2^{-t}$. Si può dimostrare [8] che la probabilità di ottenere come misura finale sul primo registro l'intero m tale che $|m - b| > \xi$ quindi *fuori* dal limite di tolleranza scelto è:

$$p(|m - b| > \xi) \leq \frac{1}{2(\xi - 1)}. \tag{2.8}$$

Per ottenere una approssimazione di ϕ con accuratezza 2^{-n} si sceglie come limite di errore $\xi = 2^{t-n} - 1$, usando la (2.8) si trova che con $t = n - p$ qubit si ha probabilità di ottenere un'approssimazione corretta che vale: $1 - 1/2(2^p - 2)$.

Per ottenere ϕ con accuratezza 2^{-n} e con probabilità di successo $P = 1 - \frac{1}{2(2^p-1)} = 1 - \epsilon$ arbitraria, basta scegliere:

$$\epsilon = \frac{1}{2(2^p - 1)} \longrightarrow t = n + \lceil \log(2 - \frac{1}{2\epsilon}) \rceil, \tag{2.9}$$

dove $\lceil x \rceil$ indica il minimo numero intero maggiore o uguale a x.

2.3.3 Le applicazioni degli algoritmi quantistici

Gli algoritmi quantistici raggiungono, in alcuni casi, delle prestazioni molto migliori rispetto ai loro corrispettivi classici nella soluzione del problema a cui sono rivolti. La scomposizione in fattori primi di un un intero a n-bit, per esempio, si esegue su un processore classico in un tempo $t \sim \Omega(N^{1/3} \log^{2/3} N)$[3] con $N = O(2^n)$, ovvero esponenziale con il numero di bit. L'algoritmo di Shor per la fattorizzazione in numeri primi [10], invece, raggiunge uno speedup esponenziale, con un tempo di risoluzione che va come $O(\log^3 n)$. Un algoritmo di ricerca quantistico presentato da Grover negli anni novanta [5], ottiene invece un aumento quadratico della velocità di risoluzione rispetto al più veloce algoritmo classico per la ricerca in un database disordinato. Questi sono solo alcuni esempi di algoritmi elaborati negli ultimi decenni.

Il machine learning è un campo di ricerca in rapida evoluzione, e viene da chiedersi in che modo la computazione quantistica possa rientrare nel futuro di questa materia. L'algoritmo HHL per la soluzione di un sistema lineare (Linear Sistem Problem), che prende il nome dalle iniziali di chi lo ha introdotto, ovvero A. W. Harrow, A. Hassidim, S. Lloyd [6], offre uno speedup esponenziale nella soluzione di questo problema, che trova applicazioni all'interno di diversi problemi inerenti al machine learning.

Un primo esempio è il cosiddetto *perceptron model*, un processo di training di machine learning che pone come obiettivo quello di separare con un iperpiano un insieme di N punti linearmente separabili. Classicamente il numero di update di training effettuati va come $O(1/\gamma^2)$ con γ il più breve tra i margini tra l'iperpiano e i punti di training. Il *quantum perceptron model*, che sfrutta l'algoritmo di ricerca di Grover, fornisce uno speedup quadratico, ovvero un tempo $t \sim O(1/\gamma)$. Un altro modello molto utilizzato di machine learning supervisionato è quello delle *support vector machines* o (SVM), usate nella analisi della regressione di serie di dati costituiti da una variabile dipendente e una o più variabili indipendenti, in cui si cerca la relazione funzionale tra questi due tipi di variabili. Per un caso particolare di SVM, basate sul metodo dei minimi quadrati, il tempo di esecuzione classico per N variabili a d dimensioni va come $O(\log \frac{1}{\epsilon} \mathrm{pol}(d, N))$ dove ϵ indica l'accuratezza. SVM quantistiche offrono uno speedup esponenziale, con un $t \sim O(\frac{1}{\epsilon} \log(dN))$.

I problemi menzionati, e molti altri, utilizzano l'algoritmo HHL o qualche altro algoritmo di soluzione di un sistema di equazioni lineari come subroutine. Nella sezione successiva si dà una descrizione precisa del problema e dell'algoritmo HHL.

[3] con la notazione Ω-grande si intende il tempo di *runtime*nel migliore dei casi, ovvero un limite inferiore al tempo di esecuzione

Capitolo 3

L'algoritmo HHL

L'algoritmo quantistico HHL, dalle iniziali di A. W. Harrow, A. Hassidim, S. Lloyd, che lo hanno introdotto nel 2009 [6], permette di risolvere il cosiddetto *Quantum Linear Sistem Problem* o QLSP, l'equivalente quantistico alla soluzione di un sistema di equazioni lineari (LSP).

Come è noto, la soluzione di un sistema lineare è direttamente riconducibile al problema di inversione di matrice: dati i vettori $\vec{x}$, $\vec{b}$ e la matrice A tali che $\mathrm{A}\vec{x} = \vec{b}$, trovare le soluzioni x_i, componenti di $\vec{x}$ equivale a trovare l'inversa A^{-1} tale che $\mathrm{A}^{-1}\vec{b} = \vec{x}$. L'algoritmo classico più noto per l'inversione di una matrice, l'algoritmo di Gauss, risolve il problema in un tempo proporzionale a $O(N^3)$, dove N è la dimensione della matrice. Il miglior algoritmo classico invece, che utilizza il metodo del gradiente coniugato [4] va in un tempo $\propto O(N \log(1/\epsilon))$ con ϵ che rappresenta la precisione del risultato.

L'algoritmo HHL valuta gli elementi della matrice inversa in un tempo che va come $O(\log(N)/\epsilon)$ ed è quindi esponenzialmente più veloce. Per sistemi di dimensione dell'ordine superiore al 10^{12}, situazioni comuni non solo in fisica, ma in qualunque ambito sia necessaria dell'attività di analisi dati, ciò porta un vantaggio considerevole.

La velocità con cui gli stati di input sono preparati e il loro errore, non sono considerati ma giocano in realtà un ruolo di fondamentale importanza, spiegato più in dettaglio nella sezione 3.2. Nelle due sezioni successive si procede a definire in modo rigoroso il problema del Sistema Lineare Quantistico (QLSP) e a descrivere le operazioni unitarie necessarie per svolgere l'algoritmo su un computer quantistico ideale, avente a disposizione un numero arbitrario di qubit.

3.1 Definizione del problema

Il *quantum linear system problem* o QLSP è così formulato [11]:

Sia data una matrice $A \in \mathbb{C}^{N \times N}$ Hermitiana con determinante unitario e due vettori $\mathbf{b}, \mathbf{x} \in \mathbb{C}^N$ tali che $\mathbf{x} := A^{-1}\mathbf{b}$. Sia poi uno stato di n qubit esprimibile come:

$$|b\rangle = \frac{\sum_i b_i |i\rangle}{|| \sum_i b_i |i\rangle ||} \text{ e } |x\rangle = \frac{\sum_i x_i |i\rangle}{|| \sum_i x_i |i\rangle ||},$$

dove b_i e x_i sono gli $i-esimi$ componenti dei vettori $\mathbf{b}$ e $\mathbf{x}$. Il Quantum Linear System Problem richiede di trovare lo stato $|\tilde{x}\rangle$ tale che $|| \, |\tilde{x}\rangle - |x\rangle \, || \leq \epsilon$ con una probabilità maggiore di $\frac{1}{2}$.

Si noti che la condizione di unitarietà del determinante può essere alleggerita sapendo che ogni matrice, se invertibile, può essere riscalata opportunamente in modo da avere $Det(A) = 1$. L'Hermiticità della matrice da invertire, invece, può essere ottenuta a partire da una matrice qualunque, si può infatti definire $\tilde{\mathrm{A}} = \begin{bmatrix} 0 & \mathrm{A}^\dagger \\ \mathrm{A} & 0 \end{bmatrix}$, che è sempre Hermitiana.

3.2 L'algoritmo

L'algoritmo HHL è composto da tre subroutine: phase estimation (PHE), valutazione dei reciproci degli autovalori (*eigenvalues inversion*) e *controlled rotations*. Al termine di queste, vengono applicate le inverse delle prime due subroutine. Uno schema del circuito è riportato in figura 3.1.

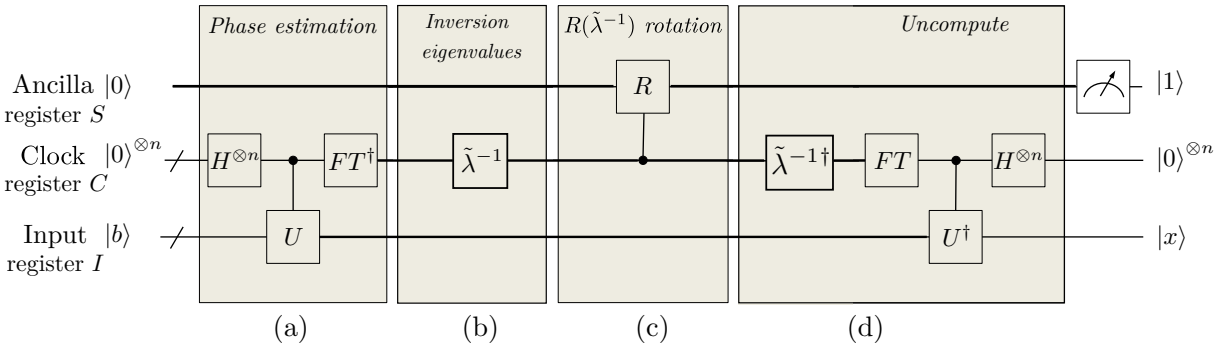


Figura 3.1: Schema del circuito che implementa l'algoritmo HHL, elaborata a partire dalla figura 5 della referenza [11]

.

Preparazione dello stato iniziale:

Data la matrice A $= \sum_j \lambda_j |u_j\rangle \langle u_j|$ si consideri il caso in cui il vettore di input $|b\rangle$ corrisponda ad un autovettore della matrice, $|u_j\rangle$, rappresentato nel circuito da un numero di qubit necessario a rappresentarlo con la precisione scelta, e che vanno a formare il cosiddetto *input register*, indicato con la lettera I in figura 3.1. Un altro registro di qubit, detto *clock register*, indicato in figura come C, presenta invece un numero t di qubit, con t dato dalla relazione (2.9), necessario quindi per rappresentare le *fasi*, o, equivalentemente, gli autovalori della matrice, come risultato della phase estimation

Quando A è Hermitiana infatti si ha che la matrice $U = e^{i\mathrm{A}}$ è unitaria, con gli stessi autovettori, e con autovalori dati da $e^{i\lambda_j} \equiv e^{i2\pi\phi}$, con λ autovalori di A. Trovare la *fase* ϕ dell'autovalore di U, allora, equivale a trovale gli autovalori di A.

Phase Estimation, PHE**:**

Come visto nella sezione 2.3.2, per valutare questo tipo di fasi possiamo applicare la Quantum Phase Estimation (parte (a) del circuito 3.1).

A partire dallo stato:

$$|0\rangle^{\otimes n} |u_j\rangle$$

otteniamo quindi lo stato:

$$|\tilde{\lambda}_j\rangle |u_j\rangle .$$

Dove $\tilde{\lambda}_j$ è la rappresentazione binaria in n bit di λ_j, che è autovalore di $|u_j\rangle$.

Eigenvalues inversion

Questa sezione (parte (b) in figura 3.1) è una parte fondamentale del circuito, poiché consiste nel valutare, senza misurare i qubit, i reciproci $\frac{1}{\lambda_j}$ degli autovalori. Sappiamo infatti che l'inverso della matrice $A = \sum_j \lambda_j |u_j\rangle \langle u_j|$ deve essere esprimibile come $A^{-1} = \sum_j \frac{1}{\lambda_j} |u_j\rangle \langle u_j|$. Il calcolo dell'inverso degli autovalori non è una procedura banale tanto che l'articolo originale [6] non lo discute. Anche in questo lavoro non tratteremo il caso generale in quanto sceglieremo una particolare matrice A per cui il reciproco degli autovalori può essere valutato in modo semplice. Rimandiamo il lettore interessato a [12] dove il circuito generale per l'inversione degli autovalori di una qualsiasi matrice A è studiato in dettaglio.

Controlled rotations:

Dopo aver calcolato l'inverso degli autovettori si applicano delle rotazioni controllate (parte (c) in fig. 3.1) tra il registro C che contiene i control qubits e su un ulteriore registro composto da un solo qubit, detto ancilla register (indicato con S in fig. 3.1). Il

j-esimo qubit di controllo di queste rotazioni, che sono in totale pari al numero di qubit del *clock register* C, rappresenta il j-esimo bit della rappresentazione binaria di $|\tilde{\lambda}_j\rangle$.

L'operatore di rotazione è:

$$R_y(\theta) = e^{-i\theta\frac{Y}{2}} = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix},$$

e agisce sullo stato $|0\rangle$ generando lo stato $\cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)|1\rangle$ sull' ancilla qubit. Dato uno stato di input $|b\rangle$ qualunque, questo può essere scritto come combinazione lineare di autostati di base $|b\rangle = \sum_j \beta_j |u_j\rangle$, per uno stato di input generico $|b\rangle$ allora, le controlled rotations generano lo stato:

$$\sum_{j=1}^{2^n} \beta_j |\tilde{\lambda}_j\rangle |u_j\rangle \left(\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle\right). \tag{3.1}$$

Scegliendo l'angolo θ delle controlled rotation come:

$$\theta = \arcsin\frac{C}{\tilde{\lambda}} \text{ con } C \text{ costante di normalizzazione,}$$

la 3.1 diventa:

$$\sum_{j=1}^{2^n} \beta_j |\tilde{\lambda}_j\rangle |u_j\rangle \left(\sqrt{1-\frac{C^2}{\tilde{\lambda}_j^2}}|0\rangle + \frac{C}{\tilde{\lambda}_j}|1\rangle\right). \tag{3.2}$$

Uncomputation

Si applicano ora le inverse di phase estimation e eigenvalue inversion, riportando il clock register nello stato di partenza $|0\rangle^{\otimes t}$, e ottenendo così lo stato complessivo:

$$|0\rangle^{\otimes t} \otimes \sum_{j=1}^{2^n} \beta_j |u_j\rangle \left(\sqrt{1-\frac{C^2}{\tilde{\lambda}_j^2}}|0\rangle + \frac{C}{\tilde{\lambda}_j}|1\rangle\right). \tag{3.3}$$

Questa operazione è riportata nella parte (d) di figura 3.1.

Misure

Dalla (3.3) si vede che nei casi in cui l'ancilla register risulti nello stato $|1\rangle$ l'input register è sicuramente nello stato $|x\rangle = A^{-1}|b\rangle$, ovvero nello stato che è soluzione del sistema lineare. Procedento allora alle misure dei due registri ancilla e di input, e considerando valide solamente quelle in cui l'ancilla è nello stato $|1\rangle$ si ottiene in I, uno stato che ha ampiezze proporzionali al vettore classico delle soluzioni.

3.2.1 Limitazioni dell'algoritmo HHL

Nonostante dalla precedente sezione il problema sembri risolto, è bene ricordare che il vettore-stato di output non può fornire tutte le informazioni che contiene in una sola volta, poiché una misura ne cambia lo stato in maniera irreparabile, fornendo solamente uno degli autostati di cui questo è sovrapposizione. La lettura diretta complessiva del risultato tramite l'iterazione di un gran numero di esecuzioni del programma richiederebbe invece un tempo $\sim O(n)$, vanificando di fatto il miglioramento esponenziali ottenuto.

Il risultato dell'algoritmo, quindi, risulta utile solo in alcuni casi particolari, ovvero in problemi in cui viene utilizzato come subroutine di algoritmi in cui sono necessari solamente dei *campioni* dal vettore delle soluzioni, oppure casi in cui sia necessario applicare un operatore sullo stato soluzione $|x\rangle$, che sono in verità la grande maggioranza.

3.3 Qiskit e IBM quantum experience

Per realizzare l'algoritmo HHL si è utilizzato *Qiskit*, un software kit scientifico open source per la computazione quantistica realizzato da IBM. Qiskit è scritto in Python e introduce un set di comandi che permettono la creazione e la manipolazione di circuiti quantistici, nonché di eseguirli su un simulatore classico locale. Attraverso il servizio *IBM Quantum experience* è poi possibile accedere da remoto a processori quantistici veri e propri messi a disposizione da IBM. Questi chip sono dotati di un numero di qubit superconduttivi che va da 5 a 14, ma nonostante il numero di qubit sia basso, non ci si deve attendere un alto grado di concordanza tra i risultati di un circuito simulato e un esecuzione reale in remoto, questo perché i processori messi a disposizione da IBM sono di tipo NISQ , ovvero computer di dimensioni intermedie ma ancora molto soggetti al fenomeno del rumore quantistico.

Altro compito di Qiskit è poi quello di ottimizzare il circuito scritto, semplificandone i gate. L'utente infatti ha a disposizione una miriade di gate logici quantistici, da quelli fondamentali presentati in sezione 1.3, a gate più complessi, anche di tipo controlled, e agenti su un numero arbitrario di qubit. Dato che un processore quantistico dispone solo di un numero limitato di gate, ad esempio nel caso di IBM sono disponibili solo le rotazioni di singolo qubit e il CNOT, Qiskit si occupa anche, dove possibile, di tradurre i gate definiti dall'utente in gate eseguibili sul processore quantistico. Questa tecnica è detta di transpiling.

Qiskit, oltre a permettere l'accesso a questi chip quantistici, provvede anche tre backend locali di simulazione di circuiti quantistici:

- Qasm Simulator, che restituisce un conteggio dei possibili risultati ottenuti dalla misura dei qubit dopo che il circuito è stato eseguito un numero dato di volte (multi-shot execution),

- Statevector Simulator, che permette, con una esecuzione single-shot, di ottenere la funzione d'onda finale dei qubit del circuito,
- Unitary Simulator, che permette l'esecuzione ideale del circuito, e restituisce l'espressione matriciale dell'operatore unitario a cui il circuito è equivalente.

In questa tesi si sono usati sia il Qasm simulator che lo Statevector simulator per testare il funzionamento dei circuiti, sia veri e propri processori quantistici.

3.4 Il circuito HHL sviluppato

Dato l'elevato grado di complessità raggiungibile da un circuito in grado di applicare l'algoritmo HHL su una matrice del tutto generale, in questo lavoro si fornisce lo sviluppo di un caso particolare, applicabile una matrice 2×2. Il circuito HHL per invertire la matrice A è stato realizzato in due varianti: la prima utilizzando un totale di 4 qubit, dei quali due sono destinati alla phase estimation, la seconda con un totale di 6 qubit, permettendo quindi una precisione nella phase estimation di due qubit in più (si riporta uno schema di quest'ultimo in figura 3.2). Obiettivo del presente lavoro è anche valutare se la precisione aggiuntiva implementata con i 2 qubit aggiuntivi produce una rilevabile differenza nel vettore soluzione del sistema lineare. Questa è l'unica differenza tra i due circuiti, che per il resto presentano tutte le componenti di un tradizionale circuito HHL:

- Quantum phase estimation, e quindi trasformata di Fourier quantistica,
- Computazione del reciproco degli autovalori λ_j della matrice,
- Controlled rotations di un angolo proporzionale a $\frac{1}{\lambda_j}$,
- Uncomputation.

3.5 Test delle componenti del circuito

Come primo obiettivo si è voluto verificare che le gli algoritmi implementati come subroutine nel circuito funzionassero correttamente. In questa sezione si riportano i risultati ottenuti dalle simulazioni della phase estimation e Fourier transform effettuate sui simulatori locali "QUASM simulator" e "Statevector simulator" (3.3), il quale permette di ricavare la funzione d'onda finale dei qubit. Questo simulatore si può utilizzare per effettuare un ulteriore controllo sulla qualità del programma implementato.

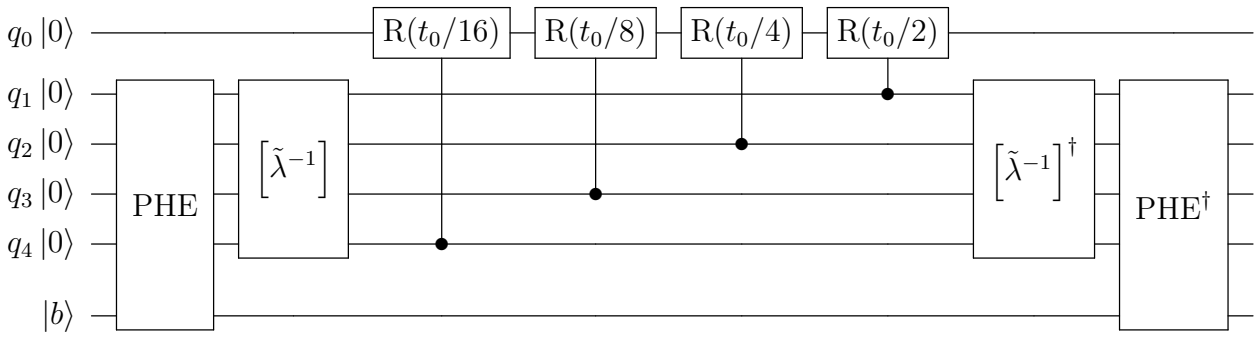


Figura 3.2: HHL, il circuito completo implementato nel caso di risoluzione della PHE utilizzando 4 qubit. Il primo qubit (q_0) è detto *ancilla register*, l'ultimo è chiamato *input register*. I qubit ($q_1 - q_4$) rimanenti, dedicati alla rappresentazione binaria degli autovalori, prendono il nome di clock register. Su di essi viene applicata l'*eigenvalue inversion* $\left[\tilde{\lambda}^{-1}\right]$.

Simulazione della Fourier transform:

Per controllare che il circuito di quantum Fourier transform sia corretto basta considerare che quando lo stato di input dei qubit è $|0\rangle^{\otimes t}$ la QFT è equivalente a una trasformata di Hadamard (2.2), e lo stato risultate è quindi $|+\rangle^{\otimes t}$. Inizializzando tutti e 4 i qubit a $|0\rangle$ ed eseguendo il circuito, ci si attende uno stato che sia quindi sovrapposizione perfettamente equilibrata di tutti i 2^4 gli stati possibili:

$$\mathrm{QFT}\,|0\rangle\,|0\rangle\,|0\rangle\,|0\rangle = |+\rangle\,|+\rangle\,|+\rangle\,|+\rangle = \frac{|0000\rangle + |0001\rangle + \cdots + |11\ldots1\rangle}{\sqrt{2^4}} = \frac{\sum_{j=0}^{2^4-1} |j_{(2)}\rangle}{\sqrt{2^4}}$$

Il circuito di quantum Fourier transform implementato in Qiskit per 4 qubit è riportato in fig. 3.3

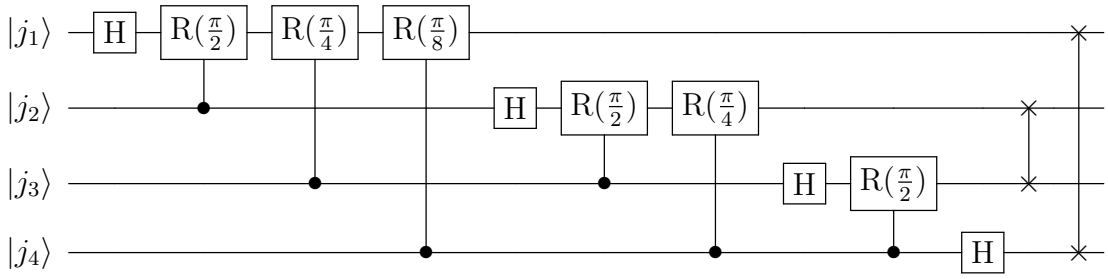
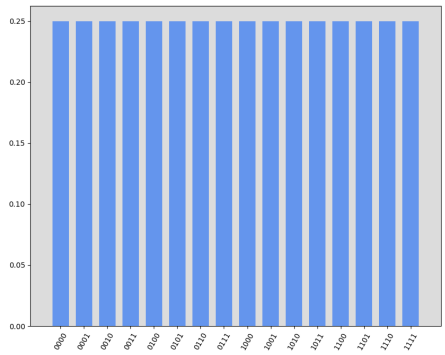
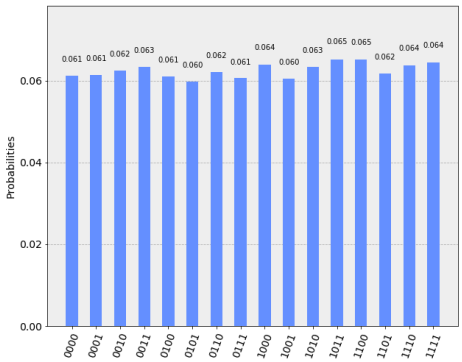


Figura 3.3: QFT. Trasformata di Fourier nel casi di 4 qubit nel registro della phase estimation.

I risultati di entrambi i simulatori, riportati in fig. 3.4, danno risultati concordi con questa previsione:



(a) "Statevector-simulator" restituisce un vettore di stato complessivo $|\psi\rangle$ sovrapposizione di tutti gli stati possibili con ampiezza uguale $|\psi\rangle = \sum_{j=0}^{2^t-1} \frac{1}{4} |j_{(2)}\rangle$.



(b) "qasm-simulator" restituisce le ampiezze di probabilità di ogni stato, ricavate dalla misura dei qubit su $20,000$ run del circuito.

Figura 3.4: Simulazione del circuito di QFT su i simulatori locali "Statevector simulator" e "QUASM simulator".

Per l'esecuzione del circuito di QFT su un vero processore quantistico si è scelta la variante con solo 2 qubit, per minimizzare gli effetti del rumore a cui i backend messi a disposizione da IBM sono soggetti. Il circuito risultate è molto semplice, se ne riporta uno schema in fig. 3.5. Per i risultati dell'esecuzione in remoto, avendo ridotto a 2 il numero di qubit, ci si aspetta uno stato finale:

$$|\psi_{exp}\rangle = \text{QFT}\,|0\rangle\,|0\rangle = |+\rangle\,|+\rangle = \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{\sqrt{2^2}} = \frac{\sum_{j=0}^{3} |j_{(2)}\rangle}{2}. \tag{3.4}$$

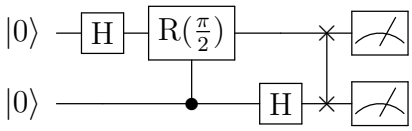


Figura 3.5: Il circuito di QFT eseguito sul processore quantistico di IBM.

Per valutare la concordanze dei risultati ottenuti e quelli teorici si può assumere, in prima approssimazione, che l'ampiezza degli stati di base che danno lo stato finale sia dato dalla radice delle probabilità ottenute con le esecuzioni sul processore.

I risultati dei run sono riportati in figura 3.6, e i valori della probabilità forniscono uno stato finale che in prima approssimazione è:

$$|\psi_{run}\rangle = \sqrt{0.288}\,|00\rangle + \sqrt{0.207}\,|01\rangle + \sqrt{0.208}\,|10\rangle + \sqrt{0.297}\,|11\rangle\,.$$

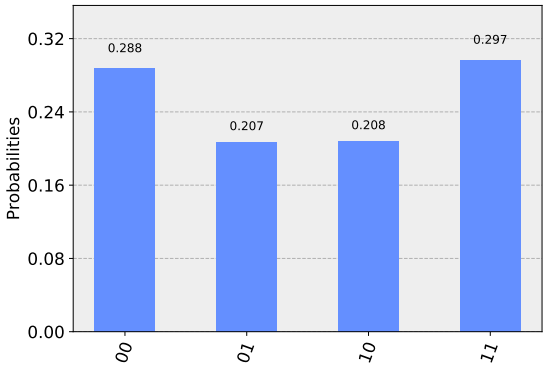


Figura 3.6: Risultati dell'esecuzione del circuito di QFT a due qubit di fig. 3.5 sul processore quantistico di IBM.

Una misura della *fidelity* f tra questo stato e lo stato atteso (3.4) sarà data dal prodotto scalare tra i due:

$$\mathrm{f} = \langle \psi_{run} | \psi_{exp} \rangle = 0.996 \tag{3.5}$$

Simulazione della quantum phase estimation:

Per quanto riguarda la quantum phase estimation, quello che si vuole valutare con l'esecuzione del circuito è la fase che compare in $e^{i\mathrm{A}} |u\rangle = e^{i\lambda} |u\rangle$, che, come spiegato in 3.2 restituisce l'autovalore λ di A.

Scegliendo

$$\mathrm{A} = \frac{1}{2} \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix},$$

che ha autovalori $\lambda_1 = 1$ e $\lambda_2 = 2$, con corrispondenti autovettori $|u_1\rangle = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ e $|u_2\rangle = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, si ha che, inizializzando l'input register con uno dei due $|u_j\rangle$, il clock register si porta nello stato che rappresenta λ_j in base binaria, ovvero 0001 per l'autovalore $\lambda_1 = 1$ e 0010 per l'autovalore $\lambda_2 = 2$. Poiché la risoluzione di 4 bit è sufficiente a rappresentare entrambi gli autovalori con assoluta precisione, ci si aspetta che il 100% degli eventi generi lo stato corretto nel clock register. Per valutare la qualità del circuito è sufficiente allora simularlo su "QUASM simulator", senza bisogno della funzione d'onda esplicita. In fig. 3.7 è riportato il circuito della PHE su 4 qubit più l'input register, così come è stato simulato, in fig. 3.8 invece si riporta il risultato ottenuto dalla simulazione per entrambi gli autovettori.

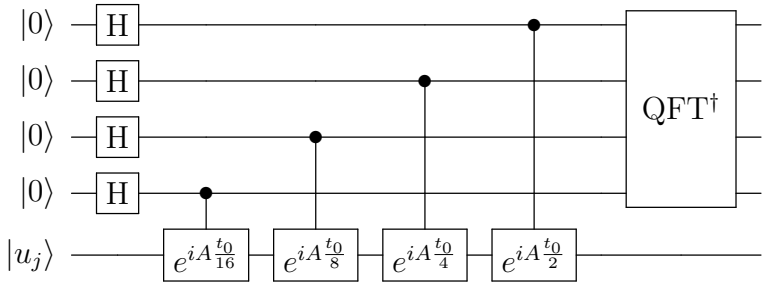
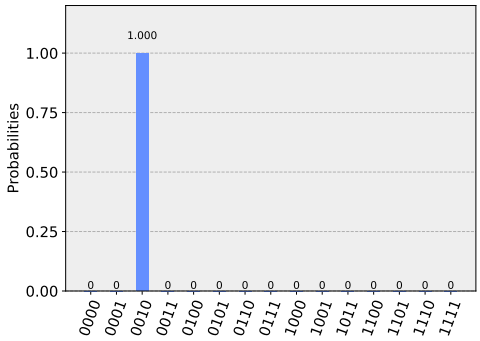
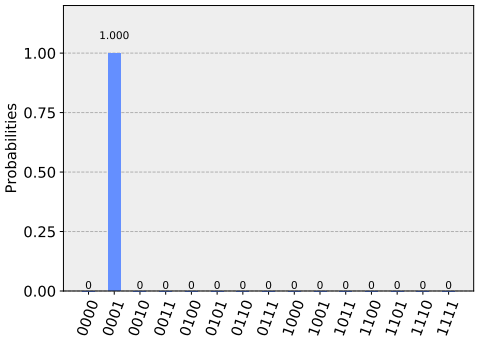


Figura 3.7: PHE, la quantum phase estimation con risoluzione di 4 qubit. L'ultimo qubit in basso è chiamato input register ed è inizializzato con autovettore della matrice A.



(a) Caso $|u_2\rangle, \lambda_2 = 2 \rightarrow 0010$ in binario



(b) Caso $|u_1\rangle, \lambda_1 = 1 \rightarrow 0001$ in binario

Figura 3.8: Simulazione del circuito di PHE sul simulatore "QUASM": per entrambi gli autovettori $|u_j\rangle$il circuito restituisce sempre lo stato corretto, che in base binaria rappresenta l'autovalore λ_j.

Per l'esecuzione della PHE su un processore quantistico vero e proprio si è fatto ricorso a una semplificazione del circuito, implementando 2 qubit nel clock register più uno per il registro di input. La procedura di *transpile* effettuata da Qiskit (3.3) per permettere l'esecuzione del circuito su un computer quantistico, non supporta gli esponenziali di matrice presenti nel circuito della PHE da 4 qubit (fig.3.7), perciò, prendendo spunto dal circuito presente in [13], si è riusciti a implementare la phase estimation e la Fourier transform per il caso della matrice A, utilizzando solo gate elementari. Il circuito riportato in fig. 3.9 è stato eseguito su un chip quantistico per entrambi gli autovettori $|u_j\rangle$, i risultati dell'esperimento sono presentati negli istogrammi di figura 3.10.

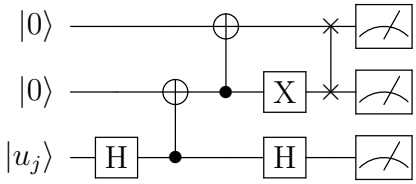
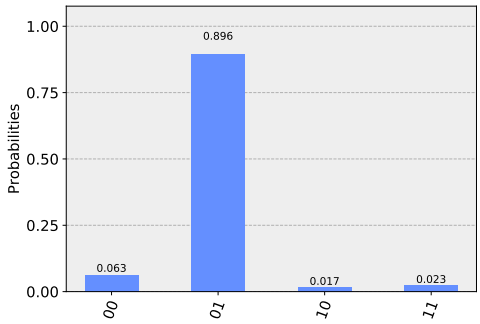
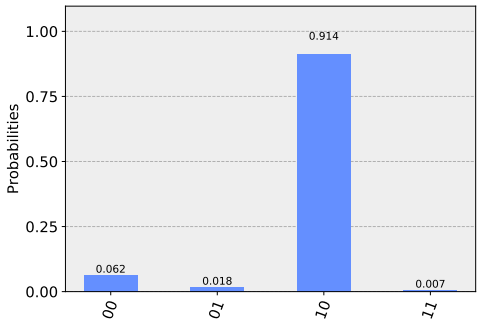


Figura 3.9: PHE, la quantum phase estimation nella versione con risoluzione di 2 qubit, così come è stato eseguito sul processore quantistico.



(a) Autovettore $|u_1\rangle$ con $\lambda = 01$



(b) Autovettore $|u_2\rangle$ con $\lambda = 10$

Figura 3.10: Risultati della PHE con risoluzione di 2 qubit eseguita sul backend quantistico di IBM per entrambi gli autovettori $|u_j\rangle$.

Anche per la Phase estimation si può usare come misura della *fidelity* dello stato finale ottenuto il prodotto scalare tra questo e lo stato finale atteso. Poiché le ampiezze teoriche degli stati che non corrispondono all'autovalore sono tutte zero, il valore della fidelity è dato semplicemente dalla radice della probabilità ottenuta nello stato corretto:

$$\text{Autovettore } |u_1\rangle \rightarrow \text{f} = \sqrt{0.896} = 0.947,$$
$$\text{Autovettore } |u_2\rangle \rightarrow \text{f} = \sqrt{0.914} = 0.956.$$

3.6 Test dell'algoritmo HHL

Test del circuito HHL con Phase estimation a 4 qubit

Una volta testate le singole componenti del circuito, è il momento di provare l'algoritmo HHL completo. Avendo fornito il modello generale in sez. 3.2, si presenta in questa sezione, un circuito per invertire la matrice A 2×2 definita nella sezione precedente. Questa matrice permette una notevole semplificazione del processo di *eigenvalues inversion* (parte (b) di fig. 3.1) tramite un design *ad hoc* del circuito.

Per valutare il reciproco degli autovalori basta uno SWAP gate. Questo SWAP deve essere applicato tra q_1 e q_2 nel clock register, nel caso di Phase estimation con precisione di 2 qubit (fig. 3.14), oppure tra il qubit q_1 e q_3 quando la PHE ha una risoluzione di 4 qubit (fig. 3.2). Per capire come calcolare l'inverso degli autovalori attraverso dei semplici SWAP gates prendiamo il caso della PHE a 4 qubit. Nel caso di autovalori uguali a 1 e 2 scambiare il primo e il terzo qubit restituisce proprio la rappresentazione binaria del reciproco di λ, modulo 2^t con t il numero di qubit nella Phase estimation, infatti:

$$0001_{(2)} \xrightarrow{\text{SWAP}_{1-3}} 0001_{(2)} = 1,$$
$$0010_{(2)} \xrightarrow{\text{SWAP}_{1-3}} 1000_{(2)} \equiv \frac{2^3}{2^4} = \frac{1}{2}.$$

Il circuito HHL con Phase estimation a 4 qubit non può essere testato sul backend quantistico di IBM, per i motivi specificati in sez. 3.5 ma si è comunque provveduto a verificare che il design del circuito complessivo fosse corretto, testando 2 diversi vettori di input:

$$|b_1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ e } |b_2\rangle = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \tag{3.6}$$

dai quali ci si attende, in output, degli stati-soluzione normalizzati $|x_j\rangle$:

$$\mathrm{A}^{-1} |b_j\rangle = \frac{|\hat{x}\rangle}{||\hat{x}||} = |x\rangle \tag{3.7}$$

$$|x_1\rangle = \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.947 \\ -0.316 \end{bmatrix} \quad |x_2\rangle = \frac{1}{\sqrt{26}} \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 0.196 \\ 0.981 \end{bmatrix}. \tag{3.8}$$

Grazie all'*uncompute* (parte (d) in fig. 3.1) del circuito, lo Statevector simulator fornisce uno stato finale dei 6 qubit che ha coefficienti uguali a zero per uno stato qualunque del clock register che non sia $|0000\rangle$. Il test allora fornisce vettori soluzione nella forma:

$$|\psi\rangle = \beta_{00} |00\rangle + \beta_{01} |01\rangle + \beta_{10} |10\rangle + \beta_{11} |11\rangle, \tag{3.9}$$

in cui il primo qubit rappresenta l'ancilla, e il secondo l'input register. Dei valori β prodotti dalla simulazione per i vettori di input $|b\rangle$, riportati in fig. 3.11 e 3.12, vanno considerate solamente quelle in cui l'ancilla qubit è nello stato $|1\rangle$ (si veda la (3.3)). Nelle figure riportate, il Most Significant Bit, ovvero l'ancilla, è quello nella posizione più a sinistra, i due coefficienti a cui siamo interessato sono quindi quelli rappresentati dalle ultime due barre.

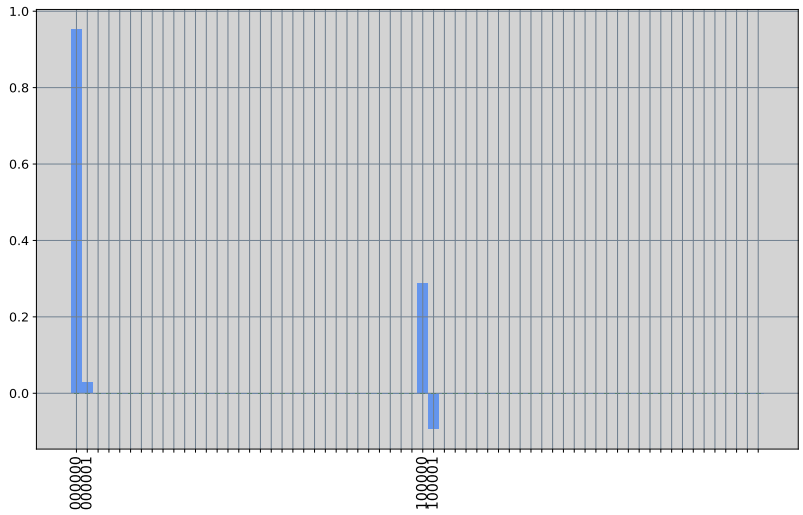


Figura 3.11: Risultati della Statevector simulator dell'HHL a 4 qubit. Per il vettore di input $|b_1\rangle$ lo stato complessivo in output è $0.952\,|00\rangle + 0.028\,|01\rangle + 0.289\,|10\rangle - 0.094\,|11\rangle$, da cui si ricava la soluzione non normalizzata $|\hat{\psi}_1\rangle = 0.289\,|10\rangle - 0.094\,|11\rangle$.

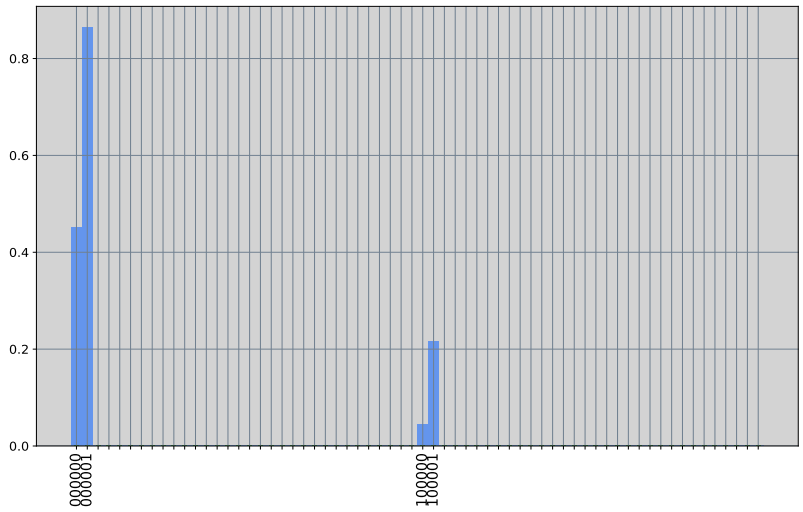


Figura 3.12: Risultati della Statevector simulator dell'HHL a 4 qubit. Per il vettore di input $|b_2\rangle$ lo stato complessivo in output $0.451\,|00\rangle + 0.864\,|01\rangle + 0.045\,|10\rangle + 0.216\,|11\rangle$. da cui si ricava la soluzione non normalizzata $|\hat{\psi}_2\rangle = 0.045\,|10\rangle + 0.216\,|11\rangle$.

Normalizzando i coefficienti degli stati $|10\rangle$ e $|11\rangle$ si ottiene:

$$\frac{\beta_j}{\sqrt{\beta_{10}^2 + \beta_{11}^2}} = \tilde{\beta}_j$$

$$|\psi_1\rangle = \tilde{\beta}_{10}\,|10\rangle + \tilde{\beta}_{11}\,|11\rangle = 0.951\,|10\rangle - 0.309\,|11\rangle = \begin{bmatrix} 0.951 \\ -0.309 \end{bmatrix}$$

$$|\psi_2\rangle = \tilde{\beta}_{10}\,|10\rangle + \tilde{\beta}_{11}\,|11\rangle = 0.204\,|10\rangle + 0.978\,|11\rangle = \begin{bmatrix} 0.204 \\ 0.978 \end{bmatrix}$$

Confrontando questi valori con i risultati esatti normalizzati (3.8) attraverso un prodotto scalare $\langle x_j|\psi_j\rangle$ si ottiene una fidelity f = 1.000 per entrambi. Il design del circuito risulta quindi corretto.

Si passa ora alla simulazione del circuito sul QUASM simulator, su quale si testa solo il vettore di input $|b_2\rangle$ poiché produce un output con ampiezze dei vettori di base tutte reali e positive. In questo modo si può assumere che la radice della probabilità ottenuta dalla misura da $n = 20'000$ run sia assimilabile ai coefficienti degli stati di base nella funzione d'onda.

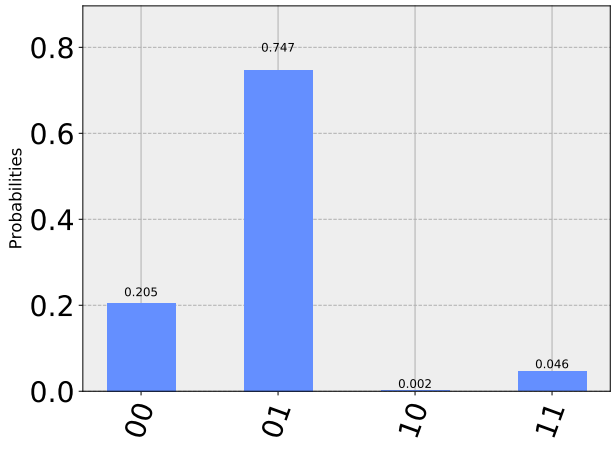


Figura 3.13: Risultati della simulazione del circuito HHL con 6 qubit totali sul simulatore QUASM nel caso del vettore di input $|b_2\rangle$. Assumendo che la radice delle probabilità ottenute sia l'ampiezza di ogni stato di base il vettore di output è $\sqrt{0.205}\,|00\rangle + \sqrt{0.747}\,|01\rangle + \sqrt{0.002}\,|10\rangle + \sqrt{0.046}\,|11\rangle$, da cui si ricava il vettore soluzione non normalizzato $\hat{\psi}_{quasm} = \sqrt{0.002}\,|10\rangle + \sqrt{0.046}\,|11\rangle$.

Procedendo alla normalizzazione degli stati in cui l'ancilla register è in $|1\rangle$ si ottiene, dai risultati riportati in figura 3.13:

$$|\psi_{quasm}\rangle = \frac{\sqrt{0.002}\,|10\rangle + \sqrt{0.046}\,|11\rangle}{0.002 + 0.046} \longrightarrow \begin{bmatrix} 0.202 \\ 0.979 \end{bmatrix}$$

Valutando nuovamente la fidelity dello stato si ottiene $\mathrm{f} = \langle x_2|\psi_{quasm}\rangle = 0.999$.

Si può quindi concludere che per la versione del circuito con Phase estimation a 4 qubit il design risulti corretto, e abbia una prestazione ottima nella soluzione del sistema lineare "A" presentato.

Test del circuito HHL con Phase estimation a 2 qubit

Per il test sul backend di IBM si è usata la versione del circuito HHL con risoluzione a due qubit riportato in figura 3.14. Anche in questo caso si è scelto come vettore di input lo stato $|b_2\rangle$. Nella prima parte della figura si riconosce la quantum phase estimation di fig. 3.9, già testata su processore quantistico, mentre prima delle controlled rotations è presente lo SWAP gate che permette di valutare il reciproco degli autovalori.

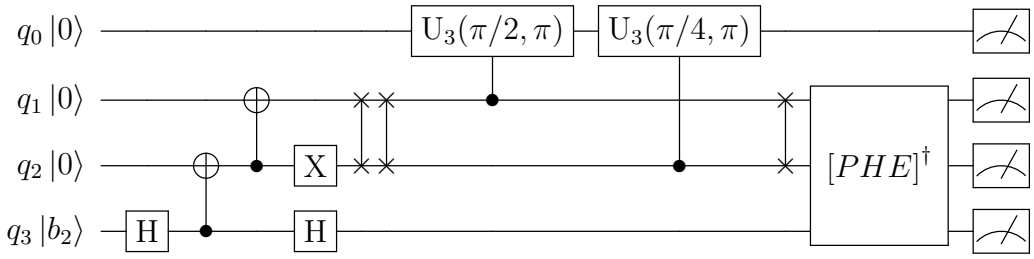


Figura 3.14: Il circuito HHL nella versione con design *ad hoc* per Phase estimation (da due qubit) e *eigenvalues inversion*.

Il test del circuito sul simulatore QUASM, ha prodotto uno stato che normalizzato diventa:

$$|\psi_{quasm}\rangle = \frac{\sqrt{0.152}\,|10\rangle + \sqrt{0.311}\,|11\rangle}{\sqrt{0.152 + 0.311}} = \begin{bmatrix} 0.573 \\ 0.819 \end{bmatrix}, \tag{3.10}$$

la cui *fidelity* risultante è $\mathrm{f} = \langle x_2|\psi_{quasm}\rangle = 0.917$.

L'esecuzione dello stesso circuito su un processore quantistico produce i risultati di figura 3.16, dai quali si estrae che lo stato complessivo nel caso di ancilla qubit nello stato $|1\rangle$, e clock register nello stato $|00\rangle$ è:

$$|\psi_{IBM}\rangle = \frac{\sqrt{0.167}\,|10\rangle + \sqrt{0.188}\,|11\rangle}{\sqrt{0.167 + 0.188}} = \begin{bmatrix} 0.689 \\ 0.728 \end{bmatrix}.$$

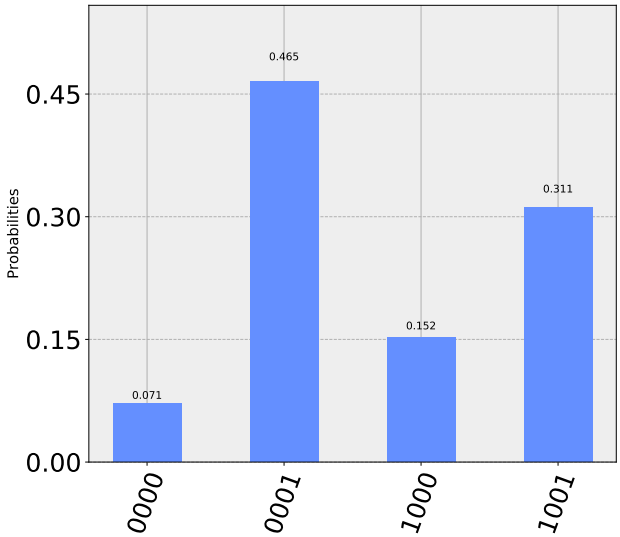


Figura 3.15: Simulazione del circuito HHL con 4 qubit totali (fig. 3.14) nel caso del vettore di input $|b_2\rangle$. Lo stato dei qubit ancilla (q_1) e di output (q_4) è $\sqrt{0.071}\,|00\rangle + \sqrt{0.465}\,|01\rangle + \sqrt{0.352}\,|10\rangle + \sqrt{0.311}\,|11\rangle$, da cui si ricava il vettore soluzione non normalizzato $\hat{\psi}_{quasm} = \sqrt{0.352}\,|10\rangle + \sqrt{0.311}\,|11\rangle$.

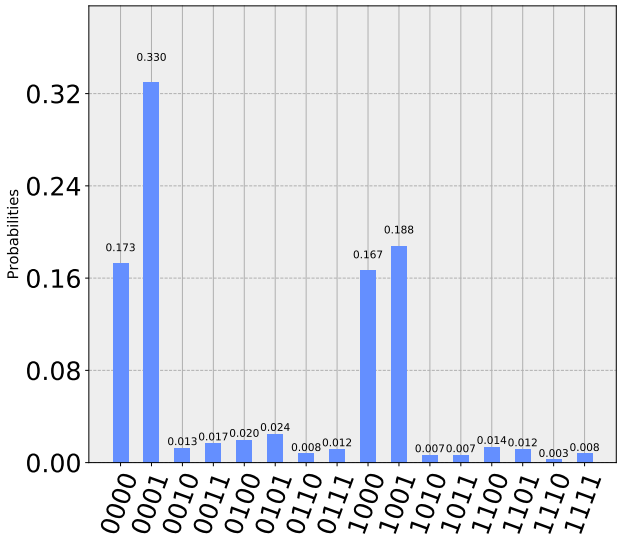


Figura 3.16: Risultati sperimentali per l'esecuzione del circuito HHL con 4 qubit totali in fig. 3.14.

Confrontando $|\psi_{IBM}\rangle$ con la soluzione $|\psi_{quasm}\rangle$ fornita dalla simulazione QUASM (3.10) si trova una *fidelity* $\mathrm{f_{IB-QA}} = 0.991$, un valore soddisfacente, che significa che per pochi qubit, il simulatore noise free approssima bene un vero chip quantistico.

La *fidelity* ottenuta rispetto allo soluzione esatta per il secondo vettore di input (3.8) è invece: $\mathrm{f_{IBM}} = 0.849$, un valore che, per quattro qubit totali, è da considerarsi buono.

Conclusioni

In questa tesi, si è analizzato un algoritmo quantistico per l'inversione di una matrice. Quest'algoritmo, introdotto in [6] da Harrow, Hassidim, e Lloyd dalle cui iniziali prende il nome, costituisce uno dei più importanti algoritmi quantistici, insieme con l'algoritmo di Shor per la fattorizzazione dei numeri primi e l'algorimo di Grover per la ricerca in un database, che presenta uno speed-up rispetto al corrispettivo classico. L'algoritmo HHL presenta al suo interno due subroutine già note in computazione qunatistica, ovvero la quantum Phase estimation (PHE), e la quantum Fourier transform (QFT).

Nella sezione 3.5 si visto che il design di queste componenti del circuito, utilizzate per ricavare gli autovalori della matrice A da invertire, è corretto. Questo è stato fatto testando le due subroutine sia su simulatori classici (fig. 3.4, 3.8) che sul backend quantistico messo a disposizione da IBM (fig. 3.6, 3.10). I risultati ottenuti dall'esecuzione su un vero processore producono una fidelity di $\mathrm{f}_{QFT} = 0.996$, e di $\mathrm{f}_{PHE} = 0.956$, valori relativamente alti se si considera la elevata percentuale di rumore sui dispositivi di IBM.

Il test dell'algoritmo HHL è stato svolto su due versioni del circuito. La prima, con quattro qubit per la parte di Phase estimation, è stata simulata sia per ottenere la funzione d'onda ideale del vettore soluzione (fig. 3.12), che simulandone diversi run con conseguenti misure dello stato finale (fig 3.13). Queste simulazioni hanno prodotto nel primo caso una fidelity pari a uno, segno che il circuito è progettato correttamente, e nel secondo caso una $\mathrm{f}_{quasm} = 0.999$.

La seconda versione dell'algoritmo HHL testata è stata quella con due qubit dedicati alla Phase estimation: di questa si è svolto un test sia sul simulatore QUASM per le misure, che sul chip di IBM vero e proprio. La fidelity prodotta in questi casi è stata di $\mathrm{f}_{quasm} = 0.917$ e $\mathrm{f}_{IBM} = 0.849$ un valore, quest'ultimo, da considerarsi buono tenendo conto del rumore quantistico a cui sono soggetti i chip di IBM, privi di sistemi di correzione degli errori.

Bibliografia

[1] J. Preskill, arXiv e-prints , arXiv:1801.00862 (2018), arXiv:1801.00862 [quant-ph] .

[2] "Ibm q experience," .

[3] F. Arute, K. Arya, R. Babbush, D. Bacon, J. Bardin, R. Barends, R. Biswas, S. Boixo, F. Brandao, D. Buell, B. Burkett, Y. Chen, J. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. M. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. Harrigan, M. Hartmann, A. Ho, M. R. Hoffmann, T. Huang, T. Humble, S. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. Martinis, Nature **574**, 505–510 (2019).

[4] J. R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, Tech. Rep. (Pittsburgh, PA, USA, 1994).

[5] L. K. Grover, in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (ACM, New York, NY, USA, 1996) pp. 212–219.

[6] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).

[7] R. P. Feynman, Int. J. Theor. Phys. **21**, 467 (1982), [,923(1981)].

[8] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. (Cambridge University Press, New York, NY, USA, 2011).

[9] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (1995).

[10] P. W. Shor, SIAM J. Comput. **26**, 1484 (1997).

[11] D. Dervovic, M. Herbster, P. Mountney, S. Severini, N. Usher, and L. Wossnig, arXiv e-prints , arXiv:1802.08227 (2018), arXiv:1802.08227 [quant-ph] .

[12] Y. Cao, A. Daskin, S. Frankel, and S. Kais, Molecular Physics **110**, 1675 (2012), arXiv:1110.2232 [quant-ph] .

[13] X. D. Cai, C. Weedbrook, Z. E. Su, M. C. Chen, M. Gu, M. J. Zhu, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, prl **110**, 230501 (2013), arXiv:1302.4310 [quant-ph] .