Università degli Studi di Milano Bicocca

Dipartimento di Matematica e Applicazioni

Corso di Laurea Triennale in Matematica

Risoluzione di sistemi lineari: Un approccio quantistico con l'algoritmo HHL

Relatore:

Prof. Lourenco Beirao Da Veiga

Correlatori:

Prof. Angelo Bassi

Giulio Crognaletti

Candidato:

Edoardo Zappia

Matricola:

866912

ANNO ACCADEMICO 2022/2023

Sommario

La risoluzione di sistemi lineari è un problema fondamentale nel mondo moderno. Settori come fisica, chimica, finanza e intelligenza artificiale sono permeati di tali sistemi perché permettono la modellizzazione di diversi scenari reali. La ricerca di algoritmi sempre più efficaci ed efficienti per risolvere questa classe di problemi risulta essere una sfida più che mai necessaria. In questa tesi si presentano due approcci completamente diversi: il metodo del gradiente coniugato, noto algoritmo classico e l'algoritmo HHL, noto algoritmo quantistico. Si presenta il confronto e le principali differenze tra computazione classica e quantistica e possibili vantaggi della seconda. Si analizza la teoria degli algoritmi e se ne testano le capacità. In questo elaborato si assume un approccio didattico: le nozioni fisiche necessarie alla trattazione sono state presentate con particolare attenzione ai loro aspetti matematici, il primo capitolo è dedicato alla spiegazione dei concetti di base della computazione quantistica, presentando alcuni esempi utili alla comprensione della parte restante della tesi. Il secondo capitolo è dedicato al metodo del gradiente coniugato, valido algoritmo classico per la risoluzione di sistemi lineari. Si presenta il metodo del gradiente per arrivare poi alla costruzione del metodo del gradiente coniugato. Si effettuano diversi test variando le dimensioni della matrice presa in esame prima su una matrice ben condizionata e poi su matrici di Hilbert. Il terzo capitolo è dedicato all'algoritmo quantistico HHL. Si presentano prima due subroutine fondamentali per diversi algoritmi quantistici e in particolare per l'HHL. Un paragrafo è dedicato alla spiegazione teorica, passo per passo, dell'algoritmo. Un paragrafo è dedicato al vantaggio teorico dell'algoritmo quantistico e delle sue limitazioni, questione di cruciale importanza in vista dello sviluppo delle tecnologie quantistiche e delle loro applicazioni. L'ultima parte della tesi è dedicata ai test.

Indice

Introduzione **1**

1 Computazione quantistica **2**

1.1 Notazione 3

1.2 Qubits 4

1.2.1 Sfera di Bloch 5

1.3 Sovrapposizione, entanglement e misura 6

1.4 Gates quantistici 9

1.4.1 Gates notevoli 10

1.4.2 Set universale 15

2 Metodo del gradiente coniugato **16**

2.1 Presentanzione del problema 16

2.2 Costruzione del metodo 16

2.2.1 Metodo del gradiente 16

2.2.2 Metodo del gradiente coniugato 18

2.3 Test dell'algoritmo 20

2.3.1 Matrice ben condizionata 20

2.3.2 Matrice mal condizionata 23

3 Algoritmo HHL **26**

3.1 Teoria preliminare 26

3.1.1 Trasformata di Fourier quantistica 26

3.1.2 Quantum Phase Estimation 31

3.2 Definizione del problema 34

3.3 Teoria 35

3.3.1 Preparazione dello stato iniziale 35

3.3.2 Phase Estimation 37

3.3.3 Rotazioni controllate 37
3.3.4 Uncomputation 38
3.3.5 Misure 39
3.4 Vantaggio teorico e limitazioni 39
3.5 Test dell'algoritmo 40
3.5.1 IBM qunatum experience e Qiskit 40
3.5.2 Premesse 41
3.5.3 Test 43

Conclusioni **48**

A Teorema spettrale complesso **49**

Elenco delle figure

1.1 Sfera di Bloch 5
1.2 Esempio di circuito quantistico 9
1.3 Applicazione di X ai vettori della base computazionale 10
1.4 X, Y, Z gates: rappresentazione circuitale 11
1.5 Applicazione di H ai vettori della base computazionale 12
1.6 Hadamard gate: rappresentazione circuitale 12
1.7 Esempio di circuito per ottenere una coppia di Bell 14

2.1 Errori per $n = 10$ nel caso di matrice ben condizionata 21
2.2 Errori per $n = 50$ nel caso di matrice ben condizionata 22
2.3 Errori per $n = 100$ nel caso di matrice ben condizionata 22
2.4 Errori per $n = 5$ nel caso di matrice mal condizionata 24
2.5 Errori per $n = 10$ nel caso di matrice mal condizionata 24
2.6 Errori per $n = 15$ nel caso di matrice mal condizionata 25

3.1 Circuito che implementa la QFT 30
3.2 Circuito che implementa la QPE 33
3.3 Schema del circuito dell'algoritmo HHL[10] 35
3.4 Circuito *ad hoc* della *QPE* 43
3.5 Circuito *ad hoc* dell'algoritmo HHL 43
3.6 Probabilità degli stati del test (qasm simulator) 44
3.7 Probabilità degli stati del test (quantum computer ibm_perth) 46

Introduzione

I computer quantistici sono radicalmente diversi dai computer classici, per questo motivo ad oggi, la loro realizzazione richiede una mole non indifferente di tempo e fondi. I computer quantistici odierni sono ancora affetti da medio/alto rumore quantistico che compromette la computazione, per questo vengono chiamati Noisy Intermediate-Scale Quantum computers (NISQ) [1]. Quando raggiungeranno un livello di precisione avanzato, queste tecnologie porteranno radicali cambiamenti nella risoluzione di alcuni problemi.

Sono già stati realizzati diversi algoritmi per questo tipo di macchine che promettono consistenti speedup rispetto alle loro controparti classiche. Peter Williston Shor, professore di Matematica Applicata all'MIT, ideò nel 1994 quello che viene chiamato Algoritmo di Shor [2]. La sua fama deriva dal fatto che, grazie all'utilizzo di fenomeni quantistici, l'algoritmo riesce a fattorizzare numeri primi con una velocità esponenzialmente maggiore rispetto al miglior algoritmo classico. Risulta essere quindi di enorme valore nella crittografia e in generale nelle telecomunicazioni che utilizzano la tecnologia RSA. Un altro algoritmo noto è l'Algoritmo di Grover [3], ideato nel 1996 dall'omonimo dottor Lov Kumar Grover, che riguarda la ricerca non strutturata. Esso porta uno speedup quadratico rispetto alla miglior controparte classica.

In questa tesi verrà presentato l'algoritmo HHL, ideato da A.W. Harrow, A. Hassidim e S. Lloyd nel 2009, per la risoluzione di sistemi di equazioni lineari.

La risoluzione dei sistemi lineari è un problema che si presenta in diversi ambiti come finanza, chimica, fisica ed ingegneria, infatti molte situazioni reali possono essere modellizzate tramite tali sistemi di equazioni. Inoltre il "Linear System Problem" (LSP) è rappresentativo per una classe di problemi detti BQP-completi che consiste nella classe di problemi risolvibili in modo efficiente con un computer quantistico. Ciò significa che ogni problema che sarà risolvibile con questo tipo di macchine, potrà essere rielaborato in termini di un sistema lineare. Vista l'importanza della risoluzione di sistemi lineari e noto che l'algoritmo HHL fornisce uno speedup esponenziale, ci si aspetta che nel futuro avrà un forte impatto applicativo.

1. Computazione quantistica

Per decenni la miniaturizzazione dei circuiti elettronici ha permesso la costruzione di computer sempre più potenti. Questo processo viene descritto dalla legge di Moore: ogni 18 mesi la densità dei transistor su un microchip e la relativa velocità di calcolo raddoppiano. Miniaturizzare i componenti presenta una forte limitazione, infatti si è arrivati a dimensioni tali per cui non possono essere trascurati gli effetti quantistici, rendendo impossibile aumentare ulteriormente la densità dei transistor e la riduzione delle dimensioni dei circuiti integrati. Proprio dalla meccanica quantistica può arrivare la soluzione con macchine radicalmente diverse. Si individua l'inizio della computazione quantistica negli anni ottanta, con la proposta del primo modello quantistico della macchina di Turing, da parte del fisico Paul Benioff. Nel 1981, durante un talk pubblicato nel 1982, Richard Phillips Feynman pronunciò la celebre frase "Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical." [4]. Il professor Feynman sottolineò, con queste parole, che per simulare la natura non possiamo usare sistemi classici ma dobbiamo utilizzare macchine che basano il loro funzionamento sulle stesse leggi della natura, cioè sulla meccanica quantistica. Una svolta della computazione quantistica avvenne nel 1994, quando il professor Peter Shor pubblicò un algoritmo in grado di fattorizzare numeri in tempo polinomiale. In questo periodo si accesero i riflettori sulla computazione quantistica, in quanto, il principale metodo di crittografia asimmetrica, chiamato RSA, si basa sul fatto che la fattorizzazione di grandi numeri richiede migliaia di anni. Quando avremo un computer quantistico fault tollerant, cioè non prono agli errori, il metodo RSA non potrà più essere considerato come sistema crittografico sicuro. Al momento della stesura di questa tesi, governi e aziende private stanno facendo una "gara" di investimenti per la costruzione di computer quantistici sempre più potenti. Le tecnologie utilizzate sono diverse e al momento non è ancora chiaro quale sia effettivamente la migliore, le principali sfruttano: superconduttori (IBM e Google); ioni (Quantinuum e Ionq); fotoni (Xanadu).

1.1 Notazione

La notazione di Dirac, nota anche come notazione bra-ket, è utilizzata in algebra lineare e nello specifico nella meccanica quantistica. Tale notazione permette di semplificare la scrittura di vettori e operatori in spazi vettoriali complessi. Ideata nel 1939 da Paul Dirac in "A New Notation for Quantum Mechanics" [5] è ora utilizzata come notazione standard nell'ambito della meccanica quantistica. Per questo motivo verrà utilizzata anche in questa tesi. Il nome "notazione bra-ket" deriva dall'utilizzo di $|v\rangle$ detto ket, che denota un vettore v in uno spazio vettoriale complesso $\mathbb{V}$ e $\langle f|$ detto bra, che denota un funzionale lineare $f : \mathbb{V} \to \mathbb{C}$

Per indicare l'agire di f su v si utilizza $\langle f|v\rangle$ Dal primo postulato della meccanica quantistica: "Ad ogni sistema fisico isolato è associato uno spazio di Hilbert chiamato spazio degli stati del sistema. Il sistema è completamente determinato da un suo vettore di stato, che è un vettore unitario nello spazio degli stati del sistema." [6], sappiamo che gli stati quantistici sono rappresentati come elementi di uno spazio di Hilbert complesso, come ad esempio lo spazio vettoriale finito-dimensionale di tutte le possibili funzioni d'onda, per questo motivo per indicare uno stato quantistico ψ utilizzeremo la notazione $|\psi\rangle$. Se lo spazio vettoriale complesso $\mathbb{V}$ ha un prodotto interno $(\cdot,\cdot)$ con primo argomento antilineare che lo rende uno spazio a prodotto interno allora $\langle\psi|$ indica il covettore di $|\psi\rangle$ appartenente al duale $\mathbb{V}^*$ di $\mathbb{V}$. Dati un vettore $|\psi\rangle$ e un funzionale $\langle\phi|$ abbiamo la corrispondenza (ϕ,ψ)=$\langle\phi|\psi\rangle$. Se consideriamo lo spazio vettoriale $\mathbb{C}^n$, possiamo identificare $|v\rangle$ con un vettore colonna e $\langle v|$ con un vettore riga. La corrispondenza tra i due consiste nel trasposto coniugato, cioè $|v\rangle$ è il trasposto coniugato di $\langle v|$ e viceversa $\langle v|^\dagger = |v\rangle$.

In questo senso ogni volta che affiancheremo i bra e i ket, e viceversa, considereremo la moltiplicazione matriciale. Dunque la notazione $\langle\psi|\phi\rangle$ indicherà un vettore risultante dalla moltiplicazione del vettore riga $\langle\psi|$ e il vettore colonna $|\phi\rangle$:

$$\langle\psi|\phi\rangle = \psi_1^*\phi_1 + \psi_2^*\phi_2 + ... + \psi_n^*\phi_n = \begin{bmatrix} \psi_1^*, \psi_2^*, ..., \psi_n^* \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix}$$

Invece la notazione $|\phi\rangle\langle\psi|$ indicherà il prodotto esterno e quindi la matrice risultante dal prodotto tra il vettore colonna $|\phi\rangle$ e il vettore riga $\langle\psi|$.

$$|\phi\rangle\langle\psi| = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} \begin{bmatrix} \psi_1^*, \psi_2^*, ..., \psi_n^* \end{bmatrix} = \begin{bmatrix} \phi_1\psi_1^* & \phi_1\psi_2^* & \dots & \phi_1\psi_n^* \\ \phi_2\psi_1^* & \phi_2\psi_2^* & \dots & \phi_2\psi_n^* \\ \vdots & \vdots & \ddots & \\ \phi_n\psi_1^* & \phi_n\psi_2^* & & \phi_n\psi_n^* \end{bmatrix}$$

1.2 Qubits

La prima differenza tra computazione quantistica e computazione classica è l'unità di misura elementare d'informazione. Nella computazione classica vengono utilizzati i bit che a livello teorico possono essere considerati come sistemi classici a due stati, caratterizzati da una certa variabile fisica. I bit possono assumere valore 0 o 1, per esempio l'assegnazione di 1 può essere determinato dal superamento di un certo valore di soglia della variabile fisica considerata. Dunque non vi sono altri valori assumibili da un bit e quindi con n bit è possibile rappresentare 2^n stati differenti.

Nella computazione quantistica si utilizzano invece sistemi quantistici a due stati, come ad esempio l'allineamento dello spin in un campo magnetico, i livelli energetici di un elettrone in un atomo o la polarizzazione di un fotone. Un qubit risulta essere quindi estremamente diverso da un bit. Il suo stato viene indicato con un vettore $|\psi\rangle$ (secondo la notazione di Dirac), appartenente ad uno spazio vettoriale complesso bidimensionale $\mathbb{C}^2$. Il sistema può assumere quindi un numero infinito di stati, ottenuti come combinazione lineare di vettori che formano una base ortonormale di autostati dell'operatore autoaggiunto associato all'osservabile del sistema. Come stati computazionali di base si scelgono $|0\rangle$ e $|1\rangle$ corrispondenti rispettivamente a $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ e $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Lo stato di un qubit può essere rappresentato nel seguente modo:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Dove α e β appartengono a $\mathbb{C}$.

Lo stato $|\psi\rangle$ è determinato dai quattro parametri reali che costituiscono α e β ma dato che $|\alpha|^2$ e $|\beta|^2$, rappresentano la probabilità di misurare rispettivamente lo stato $|0\rangle$ e lo stato $|1\rangle$ abbiamo che $|\alpha|^2 + |\beta|^2 = 1$ e questa condizione riduce i parametri a 3, vedremo nel prossimo paragrafo che in realtà i parametri necessari sono 2. Già da questo aspetto probabilistico si denota un'enorme differenza con i bit, infatti mentre lo stato di un bit può essere stabilito con assoluta certezza, non possiamo determinare con altrettanta precisione lo stato di un qubit. Solo con la misurazione e quindi facendo collassare lo stato otterremo un valore discreto, in particolare otterremo $|0\rangle$ con probabilità $|\alpha|^2$ e $|1\rangle$ con probabilità $|\beta|^2$, per questa ragione α e β sono dette ampiezze di probabilità.

1.2.1 Sfera di Bloch

Una rappresentazione grafica dello stato di un qubit è data dalla Sfera di Bloch: ogni punto della superficie della sfera rappresenta uno stato e i due stati fondamentali sono posizionati ai due poli.

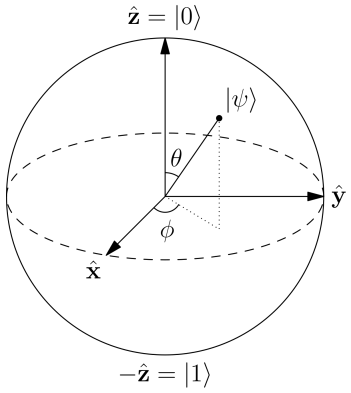

Figura 1.1: Sfera di Bloch

Questa rappresentazione risulta essere molto utile per visualizzare le operazioni su un singolo qubit, infatti esse consistono in rotazioni sulla superficie della sfera.

Pensando alla rappresentazione geometrica, un'altra formula che esprime lo stato di un qubit è la seguente:

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle)$$

Per ragioni fisiche, la fase globale non aggiunge nessuna informazione, dunque la rappresentazione diventa:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$

E il numero di parametri necessari per descrivere lo stato di un qubit diminuisce a 2.

1.3 Sovrapposizione, entanglement e misura

La meccanica quantistica è una teoria che si basa sull'algebra lineare, proprio dalla linearità e dal primo postulato possiamo esprimere uno stato come combinazione lineare di vettori di una base ortonormale, tale espressione si chiama sovrapposizione. Un esempio utilizzato prima è $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ dove $|0\rangle$ e $|1\rangle$ sono i vettori della base ortonormale e α e β sono numeri complessi chiamati ampiezze, una delle differenze fondamentali tra la computazione classica e quella quantistica è "l'utilizzo" di tali ampiezze (non probabilità). Il secondo postulato della meccanica quantistica: "L'evoluzione di un sistema (quantistico) isolato è descritta da una trasformazione unitaria. Ovvero, lo stato $|\psi_1\rangle$ al tempo $t1$ e lo stato $|\psi_2\rangle$ al tempo $t2$ sono collegati da un operatore unitario U, che dipende solo dai tempi $t1$ e $t2$, nel modo seguente $U|\psi_1\rangle = |\psi_2\rangle$" garantisce che il vettore rappresentante uno stato quantistico, rimanga unitario. Infatti l'applicazione di una matrice unitaria ad un vettore unitario, risulta in un vettore anch'esso unitario. Questo implica che la somma dei quadrati dei moduli delle ampiezze dello stato risultante è uguale a 1. Per tale motivo, come vedremo nel prossimo capitolo, le trasformazioni che vengono applicate ai qubits sono unitarie. Rimando al testo: "Quantum Computation and Quantum Information" [6] di Michael A. Nielsen e Isaac L. Chuang per una versione più raffinata del secondo postulato tramite l'equazione di Schrödinger.

Un altro ingrediente fondamentale per la computazione quantistica è l'entanglement, fenomeno puramente quantistico, non presente quindi nella meccanica classica. Definito da Einstein "spooky action at a distance", l'entanglement è stato oggetto di discussione

per anni [1]. Mi limiterò a descriverlo come fenomeno che si ha quando due o più particelle vengono generate insieme o interagiscono in un modo particolare e lo stato di una singola particella non può essere descritto indipendentemente dallo stato delle altre particelle. Questo fenomeno è estremamente contro-intuitivo perché cambiamenti dello stato di una particella in entanglement con un'altra, portano un cambiamento allo stato della seconda istantaneamente, indipendentemente dalla distanza delle due particelle. Più formalmente, considerando il quarto postulato della meccanica quantistica: "Lo spazio degli stati di un sistema composto è il prodotto tensoriale degli spazi degli stati dei componenti il sistema. Inoltre dati n sistemi preparati negli stati $|\psi_i\rangle$ lo stato composto mediante l'unione di questi n stati è $|\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle$", possiamo descrivere degli elementi in entanglement come una configurazione che non può essere espressa come prodotto tensoriale dei singoli stati che la compongono. Un esempio di due qubits in entanglement è $|00\rangle + |11\rangle$, infatti non esistono coefficienti α_1, β_1, α_2 e β_2 tali che

$$|00\rangle + |11\rangle = (\alpha_1 |0\rangle + \beta_1 |1\rangle) \otimes (\alpha_2 |0\rangle + \beta_2 |1\rangle)$$

La misura quantistica è un argomento delicato e molto differente dalla misura classica. Quest'ultima lascia il sistema inalterato, cioè nello stesso stato in cui era prima della misura, inoltre il suo risultato non varia se ripetuto più volte, a patto di mantenere le stesse condizioni. La corrispondente quantistica è invece radicalmente diversa, a partire dal fatto che una volta effettuata non si è più in grado di ricostruire lo stato in cui era il sistema prima di essa. Questa proprietà fa si che la misura sia l'unica trasformazione irreversibile utilizzata nella computazione quantistica.

A partire dai postulati della meccanica quantistica è possibile formulare il "postulato della misura" [7]: "Ogni quantità fisica misurabile, o, è descritta da un operatore Hermitiano corrispondente, O, agente sullo stato psi". Quando si va ad effettuare una misura del sistema, in realtà si sta "misurando" l'operatore Hermitiano O, corrispondente alla quantità fisica misurabile. Nella computazione quantistica si misura tendenzialmente l'osservabile Z, i cui autovettori sono i vettori della base computazionale $|0\rangle$ e $|1\rangle$. In questa

[1] Alain Aspect, John F. Clauser e Anton Zeilinger hanno vinto, nel 2022, il premio Nobel per la fisica, avendo definitivamente dimostrato che la meccanica quantistica è una teoria completa, confutando la possibilità di una teoria a variabili nascoste.

tesi verrà presentata la misura proiettiva, per avere un quadro più generale della misura quantistica rimando al testo: "Quantum Computation and Quantum Information" [6] di Michael A. Nielsen e Isaac L. Chuang. In generale una misura proiettiva è descritta da un osservabile M, un operatore Hermitiano sullo spazio degli stati del sistema che si osserva. Tale osservabile ha una decomposizione spettrale della forma

$$M = \sum_m m P_m$$

Dove P_m è la proiezione sull'autospazio di M relativo all'autovalore m. I possibili risultati della misura corrispondono agli autovalori m dell'osservabile M. Sia $|\psi\rangle$ lo stato che andiamo a misurare, la probabilità di ottenere m è data da

$$p(m) = \langle\psi|P_m|\psi\rangle$$

Se il risultato della misura è m, allora lo stato del sistema quantistico successivo alla misura è

$$\frac{P_m|\psi\rangle}{\sqrt{p(m)}}$$

Da ciò si può notare che la misura è un'operazione irreversibile perché lo stato di partenza viene "perso" e si ottiene come risultato una sua proiezione. Un altro modo per chiamare tale operazione è "misura in una base $|m\rangle$", dove $|m\rangle$ formano una base ortonormale. Ciò indica semplicemente che la misura viene effettuata con i proiettori della forma $P_m = |m\rangle\langle m|$. Come accennato in precedenza, nella computazione quantistica è molto utilizzata la misura dell'osservabile

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

che ha, evidentemente, autovalori 1 e -1 e corrispondenti autovettori rispettivamente

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

Dunque la misura di Z sullo stato $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ restituisce +1 con probabilità $\langle\psi|0\rangle\langle 0|\psi\rangle = \frac{1}{2}$ e -1 con probabilità $\langle\psi|1\rangle\langle 1|\psi\rangle = \frac{1}{2}$. In una forma più estesa:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$M_z|\psi\rangle = 1(|0\rangle\langle 0|)|\psi\rangle + (-1)(|1\rangle\langle 1|)|\psi\rangle =$$

$$1\frac{1}{\sqrt{2}}(|0\rangle\langle 0|0\rangle + |0\rangle\langle 0|1\rangle) + (-1)\frac{1}{\sqrt{2}}(|1\rangle\langle 1|0\rangle + |1\rangle\langle 1|1\rangle) =$$

$$1\frac{1}{\sqrt{2}}|0\rangle + (-1)\frac{1}{\sqrt{2}}|1\rangle$$

Quindi, rifacendoci alla forma vista in precedenza, abbiamo $\alpha = \beta = \frac{1}{\sqrt{2}}$ e dunque la probabilità di ottenere +1 risulta essere uguale alla probabilità di ottenere -1 uguale a $\frac{1}{2}$.

1.4 Gates quantistici

I computer classici sono costituiti da da circuiti elettrici contenenti cavi e porte logiche. I cavi trasportano l'informazione nel circuito mentre le porte logiche la manipolano. I computer quantistici, in modo analogo, sono costituiti da cavi e porte logiche quantistiche (gate quantistici). Uno stato quantistico entrante in un gate verrà modificato, dunque il gate rappresenta l'evoluzione temporale dello stato. Come visto nel paragrafo 1.2., le trasformazioni, per il secondo postulato, sono determinate da operatori unitari, dunque i gates quantistici possono essere espressi come matrici unitarie che agiscono sui vettori rappresentanti gli stati. I circuiti quantistici vengono spesso rappresentati come cavi a cui corrispondono i qubits e quadrati o cerchi su questi cavi corrispondenti all'azione di un operatore su quel determinato qubit.

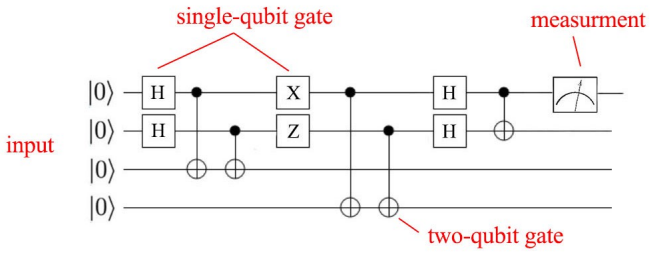

Figura 1.2: Esempio di circuito quantistico

1.4.1 Gates notevoli

Le porte logiche quantistiche possono essere classificate in base al numero di qubits su cui agiscono. Alcuni esempi possono essere gli operatori unari che agiscono su un solo qubit e quelli binari che agiscono su due qubits.

Principali gates unari

Gates X, Y e Z

Un operatore fondamentale per la computazione è il *NOT*, denotato come X, che ha la seguente forma $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, questo, agendo su uno stato, "scambia" le ampiezze associate ai vettori della base computazionale. In maniera più esplicativa, dato uno stato $|\psi\rangle$ generico della forma $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, l'operatore $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ agisce come

$$X|\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

Dunque nella misura avremo le probabilità "scambiate" rispetto allo stato di partenza. Quando agisce sui vettori della base computazionale può essere visto come un bit flip, infatti

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

e analogamente $X|1\rangle = |0\rangle$. La rappresentazione secondo la notazione di Dirac è $X = |0\rangle\langle 1| + |1\rangle\langle 0|$. Sulla sfera di Bloch, il gate *NOT* rappresenta una riflessione rispetto all'asse x:

Figura 1.3: Applicazione di X ai vettori della base computazionale

Analogamente esistono i gates Y e Z che consistono, sulla sfera di Bloch, in una riflessione rispetto all'asse y e rispetto all'asse z rispettivamente. Le rappresentazioni matriciali sono

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

X, Y e Z sono le matrici di Pauli e con la matrice identità e i loro multipli con i coefficienti ± 1 e $\pm i$ costituiscono il gruppo di Pauli. La rappresentazione circuitale di questi operatori è

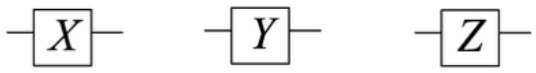

Figura 1.4: X, Y, Z gates: rappresentazione circuitale

Hadamard Gate

L'operatore cruciale della computazione quantistica è l'Hadamard gate, denotato con H. Sviluppato da John Sylvester, ma prendente il nome da Jacques Hadamard, permette di ottenere una sovrapposizione di due stati a partire da un qubit in uno stato della base computazionale. La forma matriciale è

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

dunque se si prova ad applicarlo allo stato $|0\rangle$ si ottiene

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} =$$

$$\frac{1}{\sqrt{2}} \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) = |+\rangle$$

Mentre se lo si applica allo stato $|1\rangle$ si ottiene

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} =$$

$$\frac{1}{\sqrt{2}}\left(\begin{bmatrix}1\\0\end{bmatrix}-\begin{bmatrix}0\\1\end{bmatrix}\right)=\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)=|-\rangle$$

Sia per $|+\rangle$ che per $|-\rangle$, la probabilità di misurare $|0\rangle$ è $\frac{1}{2}$ e ovviamente anche quella di misurare $|1\rangle$, l'unica differenza quindi tra i due stati è la fase relativa tra $|0\rangle$ e $|1\rangle$, infatti nel caso di $|+\rangle$ si ha che i due stati hanno la stessa fase, invece nel caso $|-\rangle$, $|0\rangle$ e $|1\rangle$ hanno una fase relativa di π. Sulla sfera di Bloch, l'Hadamard gate è rappresentabile come una rotazione di 90° rispetto all'asse y, seguita da una rotazione rispetto all'asse x di 180°:

Figura 1.5: Applicazione di H ai vettori della base computazionale

É semplice verificare che $H^2 = I$, infatti

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}=\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

dunque se applicato due volte consecutivamente si ottiene lo stato di partenza.
La rappresentazione circuitale di questo operatore è

Figura 1.6: Hadamard gate: rappresentazione circuitale

Principali gates binari

I gates binari operano su due qubits, la base computazionale dei sistemi a due qubit è fissata per convenzione a

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

CNOT gate

Gli operatori binari possono anche essere "controllati", cioè avere un qubit di controllo e un qubit obiettivo su cui agiscono solo se si verifica una determinata condizione nel qubit di controllo, il quale rimane inalterato. Un esempio di questo tipo di porta logica è il *CNOT* (controlled-NOT). Il *CNOT* applica un *NOT* al qubit obiettivo solo se il qubit di controllo è nello stato $|1\rangle$. La forma matriciale è

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Per esempio l'applicazione del *CNOT* allo stato $|10\rangle$ ha come risultato lo stato $|11\rangle$ perché il qubit di controllo è nello stato $|1\rangle$ e dunque viene effettuato un *NOT* su quello obiettivo che passa da $|0\rangle$ a $|1\rangle$, più precisamente

$$CNOT\,|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle$$

Analogamente $CNOT\,|11\rangle = |10\rangle$. Invece non abbiamo alcun effetto applicando il *CNOT* agli stati $|00\rangle$ e $|01\rangle$ perché il primo qubit, usato come controllo è nello stato $|0\rangle$ in entrambi i casi. Questo operatore permette di mettere due qubits in uno stato di entanglement,

per esempio partendo dallo stato $|00\rangle$ e applicando un Hadamard gate al primo qubit e un *CNOT* con controllo nel primo qubit e obiettivo nel secondo qubit, si ottiene lo stato $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

$$|00\rangle \to H|0\rangle|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

$$\to CNOT\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Lo stato risultante fa parte delle coppie di Bell[2].

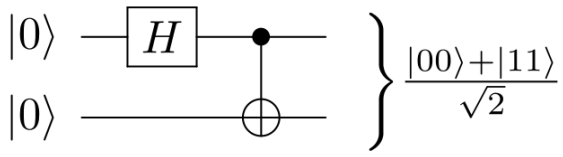

Figura 1.7: Esempio di circuito per ottenere una coppia di Bell

SWAP gate

L'operatore *SWAP* scambia i due qubits, cioè $SWAP|01\rangle = |10\rangle$ e $SWAP|10\rangle = |01\rangle$. Ovviamente gli stati $|11\rangle$ e $|00\rangle$ rimangono inalterati. La sua forma matriciale è

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

dunque

$$SWAP|01\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle$$

Analogamente per $|10\rangle$.

[2]Gli stati di Bell sono stati quantistici di due qubit che rappresentano gli esempi più semplici (e massimali) di entanglement, per maggiori informazioni rimando a https://it.wikipedia.org/wiki/Stati_di_Bell.

Ci sono anche operatori ternari che operano su tre qubits come ad esempio il *CSWAP* che consiste in uno *SWAP* di due qubits obiettivi al verificarsi di una condizione sul qubit di controllo.

1.4.2 Set universale

Un set universale di operatori è un insieme di gates con i quali è possibile costruire ogni altro tipo di operatore e quindi potenzialmente ogni tipo di algoritmo. Un set universale della computazione classica è $\{AND, OR, NOT\}$ ma anche il solo *NAND* e il solo *NOR* sono insiemi funzionalmente completi. Nella computazione quantistica un esempio di set universale è $\{CNOT, T, H\}$, dove T è l'operatore rappresentato dalla matrice $\begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$.

2. Metodo del gradiente coniugato

Il metodo del gradiente coniugato è un metodo iterativo usato per la risoluzione di sistemi lineari. Proposto nel 1952 dai matematici Magnus Hestenes e Eduard Stiefel, consiste in un perfezionamento del metodo del gradiente. Viene presentato in questa tesi perchè parte del miglior algoritmo classico per la risoluzione di sistemi lineari [8]. Tale algoritmo va in un tempo $\propto O(N\log(\frac{1}{\varepsilon}))$ con ε che rappresenta la precisione del risultato. Vedremo nel capitolo 3 che l'algoritmo HHL va come $O(\frac{\log N}{\varepsilon})$ ed è dunque esponenzialmente più veloce.

2.1 Presentanzione del problema

Il problema (LSP) viene così definito:

Dato un sistema ad N equazioni lineari, che può essere espresso nella forma $Ax = b$, dove $b \in \mathbb{R}^n$ è il vettore dei termini noti e A è una matrice $N \times N$ simmetrica e definita positiva, vogliamo trovare il vettore x che soddisfi tale relazione.

Se A non fosse simmetrica e definita positiva potremmo applicare il metodo del gradiente e il metodo del gradiente coniugato al sistema normale $A^TAx = A^Tb$ che ha la stessa soluzione del problema di partenza se A è quadrata non singolare.

2.2 Costruzione del metodo

2.2.1 Metodo del gradiente

Data A una matrice simmetrica definita positiva, consideriamo la forma quadratica

$$\phi(y) = \frac{1}{2}y^TAy - y^Tb$$

Essa ha punto di minimo dove si annulla il suo gradiente

$$\nabla\phi(y) = \frac{1}{2}(Ay) + \frac{1}{2}(y^TA) - b = \frac{1}{2}(A + A^T)y - b = Ay - b$$

$$\Rightarrow \nabla\phi(y) = 0 \Leftrightarrow Ay - b = 0 \Leftrightarrow Ay = b$$

Quindi minimizzare ϕ equivale a trovare la soluzione di $Ax = b$.

Per trovare il minimo della funzione possiamo utilizzare un metodo iterativo non stazionario del tipo

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$$

Dove α_k è la lunghezza del passo e $d^{(k)}$ la direzione di discesa.

Per determinare α_k, indipendentemente da $d^{(k)}$, imponiamo che $\frac{d}{d\alpha}\phi(x^{(k+1)}) = 0 \quad \forall d^{(k)}$

Esprimiamo in maniera più esplicita

$$\phi(x^{(k+1)}) = \frac{1}{2}(x^{(k)} + \alpha d^{(k)})^T A(x^{(k)} + \alpha d^{(k)}) - (x^{(k)} + \alpha d^{(k)})^T b =$$

$$\frac{1}{2}(x^{(k)})^T Ax^{(k)} + \frac{1}{2}(d^{(k)})^T Ad^{(k)}\alpha^2 + \frac{1}{2}(x^{(k)})^T Ad^{(k)}\alpha + \frac{1}{2}(d^{(k)})^T Ax^{(k)}\alpha - (x^{(k)})^T b - \alpha(d^{(k)})^T b =$$

$$\phi(x^{(k)}) + \frac{1}{2}(d^{(k)})^T Ad^{(k)}\alpha^2 + (d^{(k)})^T Ax^{(k)}\alpha - \alpha(d^{(k)})^T b =$$

$$\phi(x^{(k)}) + \frac{1}{2}(d^{(k)})^T Ad^{(k)}\alpha^2 - (d^{(k)})^T r^{(k)}\alpha(d^{(k)})^T Ax^{(k)}\alpha - \alpha(d^{(k)})^T b$$

Indichiamo $r^{(k)} = b - Ax^{(k)}$ il residuo al passo k-esimo

$$\Rightarrow \frac{d}{d\alpha}\phi(x^{(k+1)}) = \alpha(d^{(k)})^T Ad^{(k)} - (d^{(k)})^T r^{(k)}$$

Dunque abbiamo

$$\Rightarrow \frac{d}{d\alpha}\phi(x^{(k+1)}) = 0 \Leftrightarrow \alpha_k = \frac{(d^{(k)})^T r^{(k)}}{(d^{(k)})^T Ad^{(k)}}$$

Quindi il minimo si ottiene per

$$\alpha_k = \frac{(d^{(k)})^T r^{(k)}}{(d^{(k)})^T Ad^{(k)}}$$

Tale condizione vale anche per il metodo del gradiente coniugato, la differenza sta nella direzione di discesa $d^{(k)}$, infatti nel metodo del gradiente la direzione al passo k-esimo corrisponde a quella opposta al gradiente della funzione ϕ nel punto $x^{(k)}$

$$d^{(k)} = -\nabla\phi(x^{(k)})$$

Si può notare che tale direzione coincide con il residuo al passo k

$$-\nabla\phi(x^{(k)}) = Ax^{(k)} - b = -r^{(k)}$$

Dunque il metodo del gradiente assume la forma

$$x^{(k+1)} = x^{(k)} + \alpha_k r^{(k)}$$

2.2.2 Metodo del gradiente coniugato

É possibile dimostrare che il metodo del gradiente coniugato converge in al più n passi, dove n è la dimensione del sistema, ciò è dovuto al fatto che tale metodo utilizza una scelta più accurata della direzione di discesa.

Definizione di vettore ottimale: dato un vettore x appartenente a $\mathbb{R}^n$ e scelta una direzione p appartenente a $\mathbb{R}^n$, esso si dice ottimale rispetto alla direzione p se

$$\phi(x) \leq \phi(x+\lambda p) \quad \forall \lambda \in \mathbb{R}$$

Se x è ottimale rispetto ad ogni direzione di un sottospazio vettoriale $\mathbb{V}$ allora si dice che x è ottimale rispetto a $\mathbb{V}$.

Si può dimostrare che x è ottimale rispetto a p se e solo se la direzione p è ortogonale al residuo r cioè $p^T r = 0$.

Nel metodo del gradiente il vettore $x^{(k+1)}$ è ottimale rispetto a $d^{(k)} = r^{(k)}$, infatti

$$p^T r = (d^{(k)})^T r^{(k+1)} = (d^{(k)})^T (b - Ax^{(k+1)}) =$$

$$(r^{(k)})^T (b - Ax^{(k+1)}) = (r^{(k)})^T (b - A(x^{(k)} + \alpha_k r^{(k)})) =$$

$$(r^{(k)})^T b - (r^{(k)})^T Ax^{(k)} - \alpha_k (r^{(k)})^T Ar^{(k)} =$$

$$(r^{(k)})^T r^{(k)} - \frac{(r^{(k)})^T r^{(k)}}{(r^{(k)})^T Ar^{(k)}} (r^{(k)})^T Ar^{(k)} = 0$$

Nel metodo del gradiente coniugato la direzione $d^{(k)}$ è scelta in modo che $x^{(k+1)}$ sia ottimale rispetto ad ogni direzione precedente. Dunque il vettore $x^{(k+1)}$ è ottimale rispetto allo spazio vettoriale $\mathbb{V}$ di dimensione $k+1$, individuato dalle direzioni $d^{(i)}$ con $i = 0, \ldots, k$. Il vettore $x^{(n)}$ risulta essere ottimale rispetto all'intero spazio vettoriale e dunque coincide con il minimo cercato e la soluzione del sistema lineare perché per la definizione di ottimalità e per come abbiamo scelto le direzioni $d^{(k)}$, non esiste $y \in \mathbb{V}$ tale che

$$\phi(y) < \phi(x^{(k+1)})$$

Vediamo in dettaglio come scegliere la direzione al passo k-esimo $d^{(k)}$. Iniziamo introducendo la definizione di vettori A coniugati:

data una matrice A appartenente a $\mathbb{R}^{n\times n}$, simmetrica e definita positiva, due vettori p e q appartenenti a $\mathbb{R}^n$, si dicono A coniugati se $p^TAq = 0$.

Per scegliere la direzione $d^{(k)}$ consideriamo $x^{(k)}$ ottimale rispetto a p, quindi $p^T r^{(k)} = 0$ Sia $x^{(k+1)} = x^{(k)} + q$ e $x^{(k+1)}$ ottimale rispetto a p cioè $p^T r^{(k+1)} = 0$

$$p^T r^{(k+1)} = p^T(b - Ax^{(k+1)}) = p^T(b - Ax^{(k)} - Aq) =$$

$$p^T(r^{(k)} - Aq) = p^T r^{(k)} - p^T Aq = 0 \Leftrightarrow -p^T Aq = 0$$

cioè se e solo se p e q sono A coniugati

Scegliamo $p^{(0)} = r^{(0)}$ e $p^{(k+1)} = r^{(k+1)} - \beta_k p^{(k)}$

Per quanto visto prima occorre che

$$(p^{(k+1)})^T Ap^{(i)} = 0 \quad \text{con} \quad i = 0, \ldots, k$$

per ottenere che $x^{(k+1)}$ sia ottimale rispetto ad ogni direzione precedente.

Per soddisfare tale richiesta imponiamo

$$\beta_k = \frac{(p^{(k)})^T Ar^{(k+1)}}{(p^{(k)})^T Ap^{(k)}}$$

Quindi le iterate del metodo del gradiente coniugato sono

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

con

$$p^{(k+1)} = r^{(k+1)} - \beta_k p^{(k)}$$

$$r^{(k+1)} = b - Ax^{(k+1)}$$

$$\beta_k = \frac{(p^{(k)})^T Ar^{(k+1)}}{(p^{(k)})^T Ap^{(k)}}$$

$$\alpha_k = \frac{(r^{(k)})^T r^{(k)}}{(r^{(k)})^T Ar^{(k)}}$$

Scegliendo α_k come nel metodo del gradiente garantiamo che $x^{(k+1)}$ sia ottimale anche rispetto a $\alpha_k p^{(k)}$.

La seguente stima da un'utile caratterizzazione del comportamento del metodo del gradiente coniugato.

Teorema

Se A ha autovalori $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, si ha che

$$\|x^{k+1} - x\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right)^2 \|x^0 - x\|_A^2$$

inoltre il metodo del gradiente coniugato converge in al più n iterazioni, dove n è la dimensione del sistema lineare.

Per la dimostrazione si rimanda a [9].

2.3 Test dell'algoritmo

I test sono stati effettuati su due diverse matrici, la prima con un buon condizionamento, invece la seconda risulta essere mal condizionata. Entrambe le matrici, per quanto visto nella sezione 2.2, devono essere simmetriche e definite positive. In entrambi i casi vengono presentati i risultati ottenuti variando le dimensioni della matrice. Il test d'arresto scelto è il test basato sul residuo e la tolleranza è stata fissata a 10^{-10}.

2.3.1 Matrice ben condizionata

In questo esperimento è stata presa in esame la matrice A $n \times n$ della forma

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & \dots & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & -1 & 2 & -1 & 0 \\ 0 & \dots & \dots & \dots & 0 & -1 & 2 & -1 \\ 0 & \dots & \dots & \dots & \dots & 0 & -1 & 2 \end{bmatrix}$$

I valori di n scelti sono $n = 10$, $n = 50$ e $n = 100$. L'iterata iniziale del metodo è stata scelta casualmente con il comando $rand(n,1)$, così come la soluzione esatta. Si è calcolato l'errore ad ogni iterata k, come la distanza tra la k-esima iterata e la soluzione esatta

con $\|x_e - x_k\|$, dove x_k indica l'iterata k-esima e x_e indica la soluzione esatta del sistema. Sono stati riportati i grafici che rappresentano gli errori sia in scala normale che in scala semilogaritmica.

Caso $n = 10$

Il metodo converge alla soluzione in $k = 10$ iterate con un residuo relativo uguale a $4.3193 \cdot 10^{-32}$. La matrice ha un condizionamento pari a 48.3742.

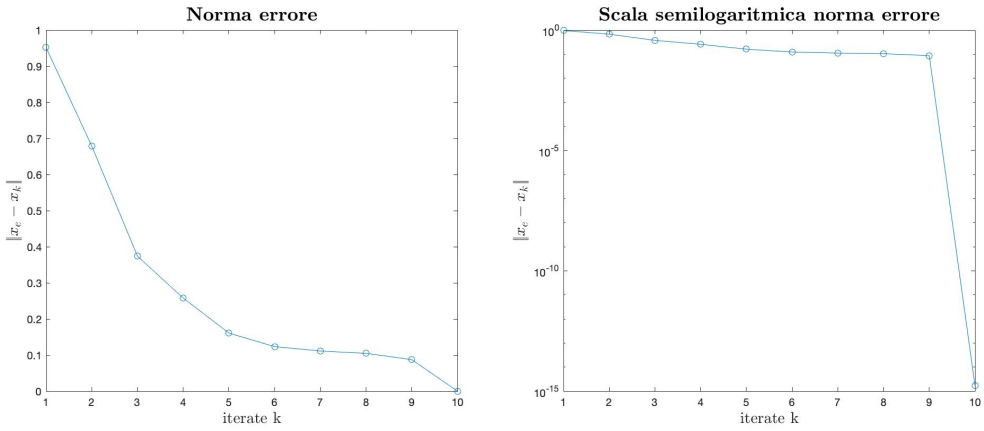

Figura 2.1: Errori per $n = 10$ nel caso di matrice ben condizionata

Caso $n = 50$

Il metodo converge alla soluzione in $k = 50$ iterate con un residuo relativo uguale a $6.3443 \cdot 10^{-30}$. La matrice ha un condizionamento pari a $1.0535 \cdot 10^{3}$.

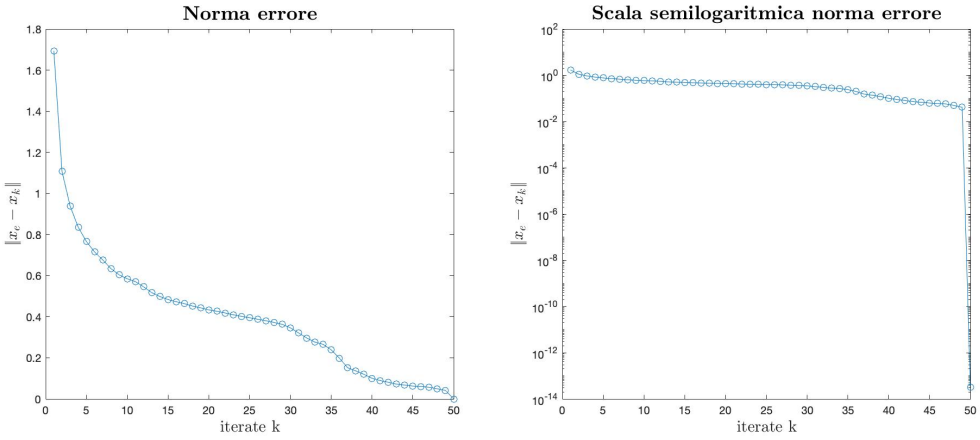

Figura 2.2: Errori per $n = 50$ nel caso di matrice ben condizionata

Caso $n = 100$

Il metodo converge alla soluzione in $k = 100$ iterate con un residuo relativo uguale a $1.8694 \cdot 10^{-27}$. La matrice ha un condizionamento pari a $4.1336 \cdot 10^{3}$.

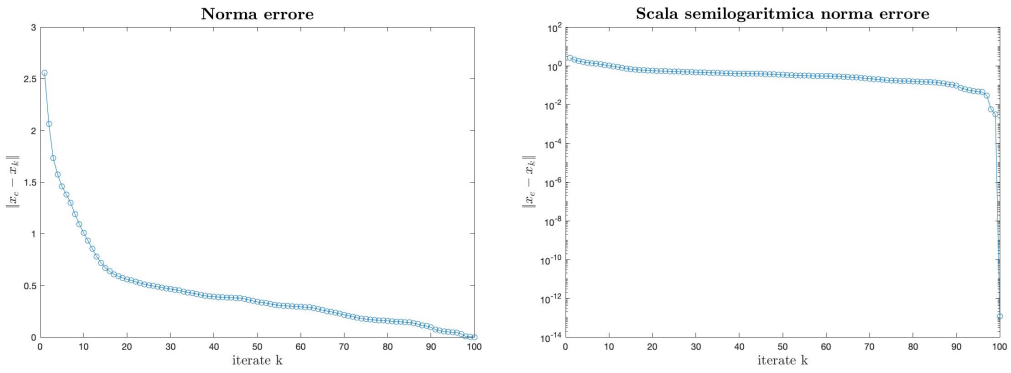

Figura 2.3: Errori per $n = 100$ nel caso di matrice ben condizionata

Conclusioni

Dai grafici che rappresentano l'errore in scala semilogaritmica si nota che aumenta notevolmente la velocità di convergenza nelle ultime iterate e l'errore diminuisce significativamente. Aumentando la dimensione della matrice si ottiene un numero di condizionamento maggiore, così come il numero di iterate necessarie per la convergenza aumenta. Si sono ottenuti ottimi risultati, mostrando l'efficacia del metodo del gradiente coniugato per ri-

solvere un sistema lineare con matrice ben condizionata, inoltre è possibile notare che il numero di passi necessari per la convergenza alla soluzione esatta, segue i risultati teorici, cioè il metodo converge in al più n passi. Si vedrà con la matrice mal condizionata che la convergenza è in realtà fortemente influenzata dall'algebra di macchina in alcuni casi.

2.3.2 Matrice mal condizionata

In questo esperimento è stata presa in esame una matrice H appartenente alla classe di matrici di Hilbert. Esse hanno entrate della forma $(h)_{ij} = \frac{1}{i+j-1}$, quindi per esempio

$$H_{3\times 3} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

Tali matrici hanno la caratteristica di essere estremamente mal condizionate, infatti il numero di condizionamento cresce in relazione alla dimensione n come $O\left(\frac{(1+\sqrt{2})^{4n}}{\sqrt{n}}\right)$, dunque cresce esponenzialmente. Per questo motivo sono state prese in esame matrici di dimensione inferiore rispetto all'esperimento precedente. Anche in questo esperimento l'iterata iniziale del metodo e la soluzione esatta sono state scelte casualmente. Inoltre sono stati riportati i grafici che rappresentano gli errori sia in scala normale che in scala semilogaritmica.

caso n = 5

Il metodo si arresta dopo k = 7 iterate con un residuo relativo uguale a $5.5159 \cdot 10^{-27}$. La matrice ha un condizionamento pari a $4.7661 \cdot 10^{5}$.

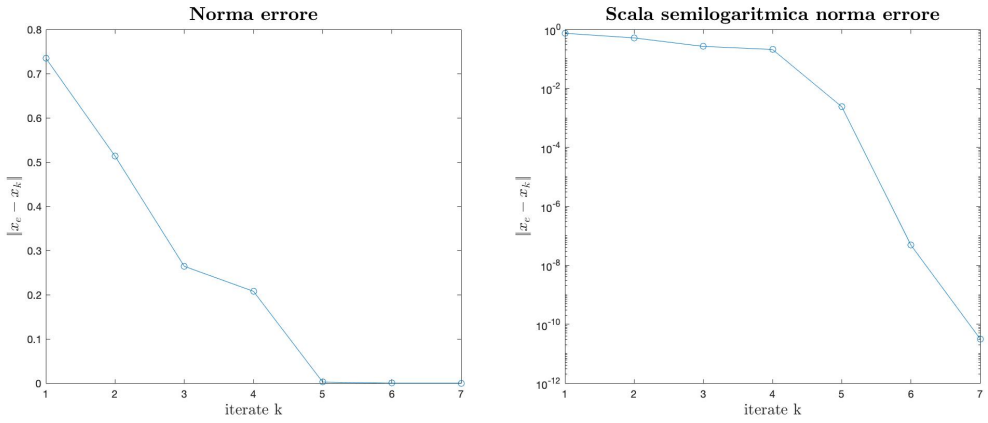

Figura 2.4: Errori per $n = 5$ nel caso di matrice mal condizionata

caso $n = 10$

Il metodo si arresta dopo $k = 17$ iterate con un residuo relativo uguale a $1.2709 \cdot 10^{-22}$. La matrice ha un condizionamento pari a $1.6025 \cdot 10^{13}$.

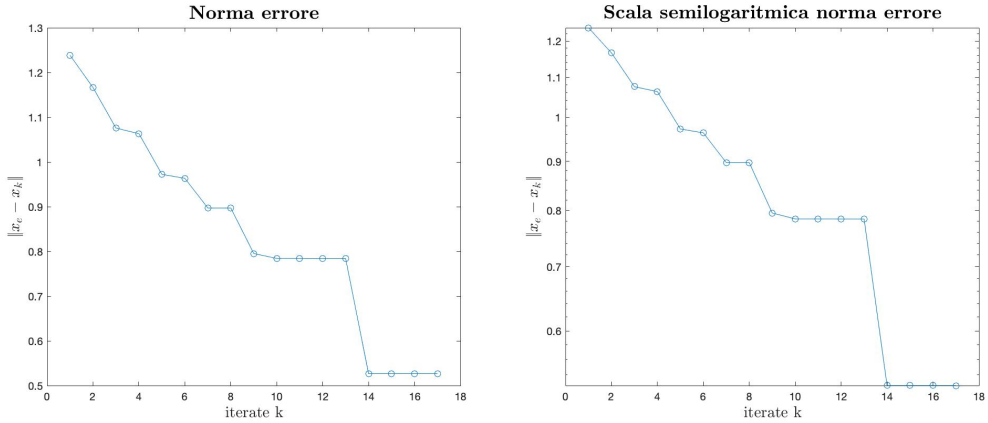

Figura 2.5: Errori per $n = 10$ nel caso di matrice mal condizionata

caso $n = 15$

Il metodo si arresta dopo $k = 19$ iterate con un residuo relativo uguale a $8.5195 \cdot 10^{-23}$. La matrice ha un condizionamento pari a $2.4960 \cdot 10^{17}$.

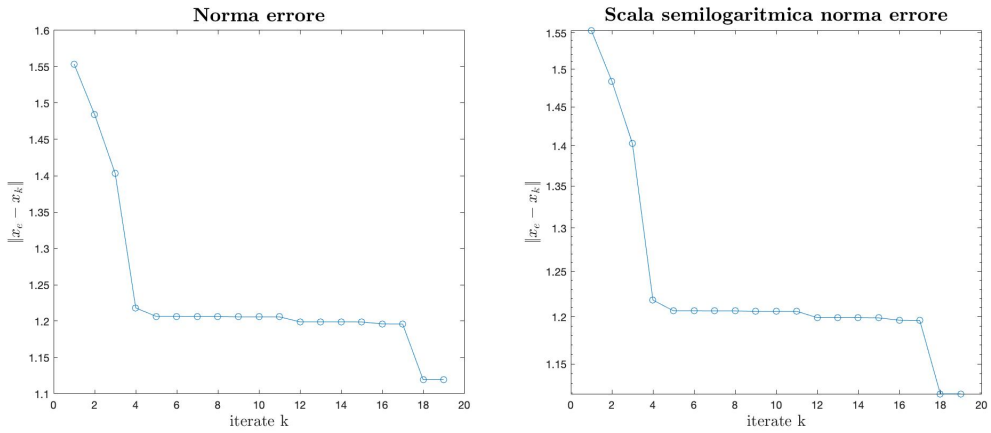

Figura 2.6: Errori per $n = 15$ nel caso di matrice mal condizionata

Conclusioni

Come si nota dai grafici, l'unico risultato soddisfacente si è ottenuto sulla matrice 5×5, infatti il metodo arriva a convergenza con un buon errore assoluto sull'ultima iterata. Dove si è utilizzato $n = 10$ e $n = 15$ sono stati ottenuti risultati poco precisi, infatti nonostante il metodo arrivi a convergenza utilizzando il test sul residuo, si nota dai grafici che la distanza tra l'ultima iterata e la soluzione esatta è dell'ordine di $1 \cdot 10^{-1}$ e 1 rispettivamente per $n = 10$ e $n = 15$. Si osserva anche che il numero di iterate necessarie per arrivare a convergenza non rispecchia quanto visto nella teoria (sez. 2.2.2), ciò è dovuto all'errore macchina e dalla propagazione degli errori causata dal mal condizionamento della matrice presa in esame; come già accennato, si nota che il numero di condizionamento cresce esponenzialmente con il crescere della dimensione. Tali risultati dimostrano che il test d'arresto sul residuo può essere fuorviante in presenza di matrici. Nonostante il metodo arrivi a convergenza, la soluzione può essere molto lontana dalla soluzione esatta.

3. Algoritmo HHL

L'algoritmo HHL, ideato nel 2009 da A. W. Harrow, A. Hassidim, S. Lloyd [10], permette di risolvere il Quantum Linear System Problem (QLSP), che consiste nell'equivalente quantistico del Linear System Problem (LSP). Tale algoritmo quantistico fa parte di una classe molto particolare, infatti porta uno speedup esponenziale rispetto al miglior algoritmo classico per la risoluzione di sistemi lineari. La questione diventa interessante quando la matrice che compone il sistema diventa di grandi dimensioni, per esempio con N dell'ordine di 10^{12}, infatti il miglior algoritmo classico va in un tempo $\propto O(N\log(\frac{1}{\varepsilon}))$ con ε che rappresenta la precisione del risultato, invece l'algoritmo HHL va come $O(\frac{\log N}{\varepsilon})$. Sistemi di equazioni lineari di queste dimensioni sono comuni sia in fisica che nell'analisi dati, si prospetta quindi che l'algoritmo trattato in questo capitolo sarà di grande utilità in diversi ambiti.

3.1 Teoria preliminare

Prima di analizzare l'algoritmo HHL, vediamo due componenti che saranno fondamentali per lo studio successivo.

3.1.1 Trasformata di Fourier quantistica

La trasformata di Fourier, sviluppata dal matematico francese Jean Baptiste Joseph Fourier nel 1822, è una trasformata integrale, cioè un operatore che tramite l'integrazione trasforma la funzione su cui è applicato in un'altra funzione. La sua importanza deriva dal fatto che permette di scrivere una funzione dipendente dal tempo, come combinazione lineare (eventualmente continua) di funzioni di base esponenziali. In altre parole la trasformata di Fourier associa ad una funzione i valori dei coefficienti di questi sviluppi lineari, rappresentandola quindi nel suo spettro cioè nel dominio delle frequenze. Per questo ha molte applicazioni sia nell'ambito fisico che in quello informatico.

In ambito discreto la trasformata di Fourier è detta DFT e agisce su un vettore N-dimensionale come una matrice quadrata D di dimensione $N \times N$ avente come colonne vettori che formano una base ortonormale, chiamata base di Fourier, che si ottengono applicando la DFT ai vettori della base canonica. Avremo dunque che le entrate di D sono

$$D_{jk} = \frac{1}{\sqrt{N}} e^{\frac{i2\pi jk}{N}}$$

La trasformata di Fourier quantistica, in modo analogo, è definita dalla sua applicazione sui vettori della base computazionale di un sistema N-dimensionale

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

con $0 \leq j \leq N-1$, $j \in \mathbb{Z}$ e $|k\rangle$ vettore della base computazionale. Spesso viene indicato $e^{\frac{2\pi i}{N}}$ con ω, dunque l'espressione precedente può essere scritta come

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} |k\rangle$$

Supponiamo di avere un vettore N-dimensionale $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$ tale che lo stato quantistico $|\psi\rangle$ venga rappresentato come

$$|\psi\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$$

La QFT dello stato $|\psi\rangle$ è data da

$$|\Phi\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} e^{\frac{i2\pi jk}{N}} x_j |k\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$$

con y_k definita da $y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{i2\pi jk}{N}} x_j$ e $k = 0, \ldots, N-1$

Proviamo a ricavare la matrice corrispondente all'operatore QFT. Per farlo partiamo dalla relazione

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{\frac{i2\pi jk}{N}} x_j = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega^{jk} x_j$$

Possiamo scrivere equivalentemente:

$$
\begin{aligned}
y_0 &= \frac{1}{\sqrt{N}}(x_0 + x_1 + x_2 + \cdots + x_{N-1}) \\
y_1 &= \frac{1}{\sqrt{N}}(x_0 + x_1\omega + x_2\omega^2 + \cdots + x_{N-1}\omega^{N-1}) \\
y_2 &= \frac{1}{\sqrt{N}}(x_0 + x_1\omega^2 + x_2\omega^4 + \cdots + x_{N-1}\omega^{2N-2}) \\
&\vdots \\
y_{N-1} &= \frac{1}{\sqrt{N}}(x_0 + x_1\omega^{N-1} + x_2\omega^{2N-2} + \cdots + x_{N-1}\omega^{(N-1)(N-1)})
\end{aligned}
$$

Dunque la matrice che rappresenta l'operatore QFT è

$$
\frac{1}{\sqrt{N}}\begin{bmatrix}
1 & 1 & 1 & 1 & \dots & 1 \\
1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\
1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2N-2} \\
1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3N-3} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & \omega^{N-1} & \omega^{2N-2} & \omega^{3N-3} & \dots & \omega^{(N-1)(N-1)}
\end{bmatrix}
$$

Tale matrice risulta essere unitaria dunque l'operatore può essere utilizzato nella computazione quantistica. Ricaviamo il circuito che implementa la quantum Fourier transform. Per prima cosa sottolineiamo che dato lo stato $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, l'applicazione su di esso risulta essere $QFT|\psi\rangle = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle$. Dunque applicare la QFT a un solo qubit equivale ad applicare il gate di Hadamard. Consideriamo ora un sistema a n-qubit con $N = 2^n$. Sia $|j\rangle$ un vettore della base con $j = 0, \dots, N-1$. Useremo la rappresentazione binaria data da

$$
j = j_1 j_2 \dots j_n \quad \text{con} \quad j_i = \{0, 1\}
$$

$$
j = j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_n 2^0 = 2^n \sum_{l=1}^{n} 2^{-l} j_l
$$

Utilizzando questa notazione per rappresentare lo stato ottenuto dall'applicazione di QFT

a $|j\rangle$ si ottiene

$$|j\rangle = |j_1 j_2 \cdots j_n\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |k\rangle$$

rimpiazzando k usando l'equazione precedente

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{o=1}^{n} k_o 2^{-o}\right)} |k_1 k_2 \cdots k_n\rangle$$

$$= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \prod_{o=1}^{n} e^{2\pi i j k_o 2^{-o}} |k_1 k_2 \cdots k_n\rangle$$

scrivendo lo stato quantistico come prodotto tensoriale di n qubits

$$= \frac{1}{2^{n/2}} \otimes_{o=1}^{n} \sum_{k_o=0}^{1} e^{2\pi i j k_o 2^{-o}} |k_o\rangle$$

$$= \frac{1}{2^{n/2}} \otimes_{o=1}^{n} \left(|0\rangle + e^{2\pi i j 2^{-o}} |1\rangle\right)$$

$$= \frac{1}{2^{n/2}} \left(\left(|0\rangle + e^{2\pi i j 2^{-1}} |1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i j 2^{-2}} |1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i j 2^{-n}} |1\rangle\right)\right)$$

Per capire quali gate usare nel circuito ci basiamo su due fattori: applicare la quantum Fourier transform a un qubit equivale ad applicare un gate di Hadamard e le fasi associate allo stato finale ottenuto nell'espressione precedente sono della forma $e^{\frac{2\pi i}{2^k}}$ con $k = 1, \ldots, n$. Useremo quindi due gate: l'Hadamard gate e una rotazione controllata (CR_k). La rotazione è espressa in forma matriciale da

$$CR_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$$

Tale operatore applica allo stato quantistico una fase relativa di $e^{\frac{2\pi i}{2^k}}$ se entrambi i qubit di controllo sono nello stato $|1\rangle$. Sulla sfera di Bloch corrisponde ad una rotazione attorno all'asse z.

$$CR_k|00\rangle \rightarrow |00\rangle$$

$$CR_k|01\rangle \rightarrow |01\rangle$$

$$CR_k|10\rangle \rightarrow |10\rangle$$

$$CR_k|11\rangle \rightarrow e^{\frac{2\pi i}{2^k}} |11\rangle$$

Sia $|j_1 j_2 \dots j_n\rangle$ lo stato in input. Applicando l'Hadamard gate al primo qubit si ottiene

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i j_1 2^{-1}}|1\rangle)|j_2 \dots j_n\rangle$$

dato che $e^{\pi i j_1} = 1$ se $j_1 = 0$ e $e^{\pi i j_1} = -1$ se $j_1 = 1$. Applichiamo ora l'operatore CR_2 con il primo qubit come obiettivo e il secondo come controllo. Lo stato risultante è

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(j_1 2^{-1} + j_2 2^{-2})}|1\rangle)|j_2 \dots j_n\rangle$$

Dopo aver applicato CR_i con l'i-esimo qubit come controllo e il primo come obiettivo consecutivamente per $i = 3, \dots, n$, si ottiene lo stato

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(j_1 2^{-1} + j_2 2^{-2} + j_3 2^{-3} + \dots + j_n 2^{-n})}|1\rangle)|j_2 \dots j_n\rangle$$

Si noti che $j = 2^n(j_1 2^{-1} + j_2 2^{-2} + j_3 2^{-3} + \dots + j_n 2^{-n})$ quindi si ottiene

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i j 2^{-n}}|1\rangle)|j_2 \dots j_n\rangle$$

Ripetendo la stessa procedura usando il secondo qubit come obiettivo, cioè applicando un gate di Hadamard e CR_i con il $i+1$-esimo qubit come controllo e il secondo come obiettivo con $i = 2, \dots, n-1$, si ottiene lo stato

$$\frac{1}{2^{\frac{1}{4}}}(|0\rangle + e^{2\pi i j 2^{-n}}|1\rangle)(|0\rangle + e^{2\pi i j 2^{-n+1}}|1\rangle)|j_3 \dots j_n\rangle$$

Ripetendo la medesima procedura per ogni qubit lo stato risultante è

$$\frac{1}{2^{\frac{n}{2}}}(|0\rangle + e^{2\pi i j 2^{-n}}|1\rangle) \otimes (|0\rangle + e^{2\pi i j 2^{-n+1}}|1\rangle) \otimes \dots \otimes (|0\rangle + e^{2\pi i j 2^{-1}}|1\rangle)$$

Si noti che questo stato corrisponde a quello trovato nell'analisi precedente, tranne per il fatto che i qubit sono nell'ordine inverso. Tale ordine può essere modificato tramite lo *SWAP* gate.

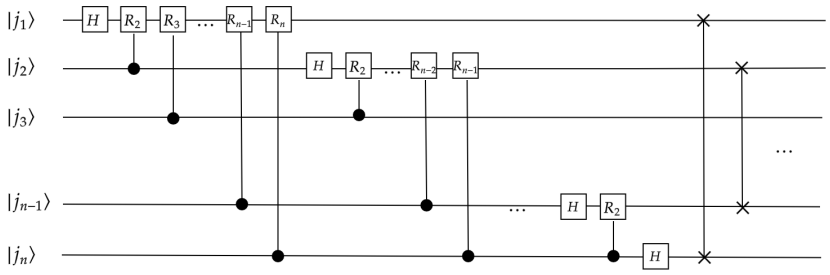

Figura 3.1: Circuito che implementa la QFT

Per completezza viene esposta brevemente anche l'inversa della trasformata di Fourier quantistica. Indicata con $QFT^\dagger$, soddisfa la relazione $QFT \cdot QFT^\dagger = I$ ed è definita da

$$QFT^\dagger |k\rangle = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} e^{-\frac{2\pi i jk}{N}} |l\rangle$$

Per implementare $QFT^\dagger$ basta applicare in ordine inverso tutti gli operatori utilizzati per implementare QFT.

3.1.2 Quantum Phase Estimation

Un algoritmo fondamentale come subroutine in molti algoritmi quantistici è la Quantum Phase Estimation (QPE), tale procedimento permette, sfruttando la QFT, di ottenere gli autovalori di un operatore unitario. Più precisamente, dato un operatore unitario U, la QPE permette di ottenere con un'arbitraria approssimazione gli autovalori λ_i di tale operatore. Il nome deriva dal fatto che, essendo U unitario è possibile esprimerlo con e^{iA} e i suoi autovalori risultano essere fasi complesse della forma $\lambda_i = e^{2\pi i \phi_i}$.

Sia U un operatore unitario con autovettore $|\psi\rangle$ associato all'autovalore $e^{2\pi i\phi}$, dunque $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$. L'obiettivo è stimare ϕ.

Introduciamo una versione controllata dell'operatore U, che chiameremo CU. L'azione di CU su due qubit è data da

$$CU(|0\rangle|\psi\rangle) \to |0\rangle|\psi\rangle \quad \text{e} \quad CU(|1\rangle|\psi\rangle) \to e^{2\pi i\phi}|1\rangle|\psi\rangle$$

Quando il primo qubit è in sovrapposizione degli stati della base computazionale, l'operatore CU pone una fase di $e^{2\pi i\phi}$ allo stato $|1\rangle$:

$$CU\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}|\psi\rangle\right) \to \frac{|0\rangle+e^{2\pi i\phi}|1\rangle}{\sqrt{2}}|\psi\rangle$$

Quindi l'applicazione ad uno stato arbitrario è data da:

$$\alpha|0\rangle|\psi\rangle+\beta|1\rangle|\psi\rangle \to \alpha|0\rangle|\psi\rangle+e^{2\pi i\phi}\beta|1\rangle|\psi\rangle = (\alpha|0\rangle+e^{2\pi i\phi}\beta|1\rangle)|\psi\rangle$$

Introduciamo anche le "potenze" dell'operatore U, e quindi di CU, che denoteremo con U^{2^j}, il risultato della loro applicazione è dato da:

$$
\begin{aligned}
&U^{2^0}|\psi\rangle = U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle \\
&U^{2^1}|\psi\rangle = UU|\psi\rangle = Ue^{2\pi i\phi}|\psi\rangle = e^{2\pi i 2^1\phi}|\psi\rangle \\
&\vdots \\
&U^{2^j}|\psi\rangle = e^{2\pi i 2^j\phi}|\psi\rangle
\end{aligned}
$$

Per implementare l'algoritmo abbiamo bisogno di due registri, il primo conterrà t qubit nello stato $|0\rangle$ e il secondo conterrà $|\psi\rangle$. Supponiamo di numerare i qubit del primo registro da 1 a t. t dipende dal numero di cifre di precisione e dalla probabilità di successo nello stimare ϕ, più avanti verranno dati maggiori dettagli. Iniziamo applicando un Hadamard gate a tutti i qubit del primo registro, dunque otterremo lo stato

$$\frac{1}{2^{\frac{t}{2}}}(|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle)|\psi\rangle$$

Applichiamo un CU^{2^j} con il qubit $t-j$ come controllo con $j = 0, \ldots, t-1$.
$j = 0 \rightarrow CU^{2^0}$ con il qubit t come controllo

$$\frac{1}{2^{\frac{t}{2}}}(|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + e^{2\pi i\phi 2^0}|1\rangle)|\psi\rangle$$

$j = 1 \rightarrow CU^{2^1}$ con il qubit $t-1$ come controllo

$$\frac{1}{2^{\frac{t}{2}}}(|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i\phi 2^1}|1\rangle) \otimes (|0\rangle + e^{2\pi i\phi 2^0}|1\rangle)|\psi\rangle$$

$$\vdots$$

$j = t-1 \rightarrow CU^{2^{t-1}}$ con il qubit 1 come controllo

$$\frac{1}{2^{\frac{t}{2}}}(|0\rangle + e^{2\pi i\phi 2^{t-1}}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i\phi 2^1}|1\rangle) \otimes (|0\rangle + e^{2\pi i\phi 2^0}|1\rangle)|\psi\rangle$$

Come si può notare dalla formula appena ottenuta, abbiamo codificato ϕ nella fase dello stato $|1\rangle$ per ogni qubit del primo registro. Esprimiamo il risultato appena trovato in

una diversa formulazione

$$\frac{1}{2^{\frac{t}{2}}}\bigotimes_{l=1}^{t}(|0\rangle + e^{2\pi i\phi 2^{t-l}}|1\rangle)|\psi\rangle =$$
$$\frac{1}{2^{\frac{t}{2}}}\bigotimes_{l=1}^{t}\sum_{k_l=0}^{1} e^{2\pi i k_l \phi 2^{t-l}}|k_l\rangle|\psi\rangle =$$
$$\frac{1}{2^{\frac{t}{2}}}\sum_{k_1=0}^{1}\cdots\sum_{k_t=0}^{1} e^{2\pi i\sum_{l=1}^{t} k_l 2^{t-l}\phi}|k_1\ldots k_t\rangle|\psi\rangle =$$
$$\frac{1}{2^{\frac{t}{2}}}\sum_{k=0}^{2^t-1} e^{2\pi ik\phi}|k\rangle|\psi\rangle$$

Se ϕ fosse esprimibile usando esattamente t bit nella forma $\phi = 0.\phi_1\ldots\phi_t = \frac{\phi_1\ldots\phi_t}{N} = x/N$ con $N = 2^t$, potremmo scrivere il primo registro come

$$\frac{1}{2^{\frac{t}{2}}}\sum_{k=0}^{2^t-1} e^{\frac{2\pi ixk}{N}}|k\rangle$$

Tale risultato è esattamente l'applicazione della QFT ai vettori della base $|j\rangle$

$$\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{i2\pi jk}{N}}|k\rangle$$

Applicando l'inversa della QFT ($QFT\dagger$) misuriamo $|x\rangle|\psi\rangle = |\phi_1\ldots\phi_t\rangle|\psi\rangle$, dunque otteniamo un'approssimazione di ϕ con t bit.

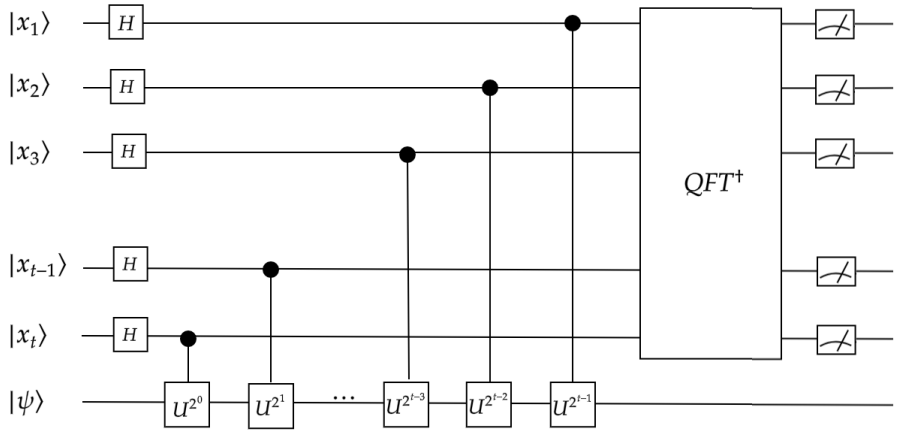

Figura 3.2: Circuito che implementa la QPE

Precisiamo ora quale è il numero t di qubit necessari per stimare ϕ con precisione caratterizzata dall'intero positivo γ.

Sia y l'intero tale che $\frac{y}{2^t} = 0.y_1 \dots y_t < \phi$ è l'approssimazione più vicina e minore di ϕ in t-bit, la differenza tra ϕ e y è $\phi - \frac{y}{2^t} \leq 2^{-t}$. Si può dimostrare [7] che la probabilità di ottenere come misura finale sul primo registro l'intero m tale che $|m-y| > \gamma$ quindi fuori dal limite di tolleranza scelto è:

$$p(|m-y| > \gamma) \leq \frac{1}{2(\gamma-1)}.$$

Scegliendo come limite di errore $\gamma = 2^{t-n} - 1$, si ottiene un'approssimazione di ϕ con accuratezza 2^{-n}. Inoltre tramite la formula precedente si ottiene che con $t = n - p$ qubit, la probabilità di avere un'approssimazione corretta è $1 - \frac{1}{2(2^p-2)}$. Sia $\varepsilon = \frac{1}{2(2^p-2)}$, per ottenere ϕ con accuratezza 2^{-n} e con probabilità $P = 1 - \varepsilon$ arbitraria, basta prendere

$$t = n + \lceil \log(2 - \frac{1}{2\varepsilon}) \rceil$$

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

Torniamo ora all'algoritmo HHL e mostriamo come vengono applicate le subroutines appena viste.

3.2 Definizione del problema

Il Quantum Linear System Problem o QLSP è così definito:

Data una matrice Hermitiana $A \in \mathbb{C}^{N \times N}$ e due vettori b, x appartenenti a $\mathbb{C}^N$ tali che $x = A^{-1}b$. Sia uno stato di n qubit esprimibile come:

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

dove b_i e x_i sono le $i-esime$ componenti dei vettori b e x. Il Quantum Linear System Problem consiste nel trovare lo stato $|\tilde{x}\rangle$ tale che $\||\tilde{x}\rangle - |x\rangle\| \leq \varepsilon$ con probabilità maggiore di $\frac{1}{2}$

Si noti che a partire da una matrice A qualsiasi, si può definire $\tilde{A} = \begin{bmatrix} 0 & A^\dagger \\ A & 0 \end{bmatrix}$ che risulta essere Hermitiana, infatti

Definizione di matrice Hermitiana

Una matrice A si dice Hermitiana se essa coincide con la matrice trasposta della sua complessa coniugata, cioè se $A = A^\dagger$

3.3 Teoria

L'algoritmo HHL può essere diviso in tre subroutine: phase estimation (PHE), rotazioni controllate (controlled rotations) e l'inversa della phase estimation (Uncompute).

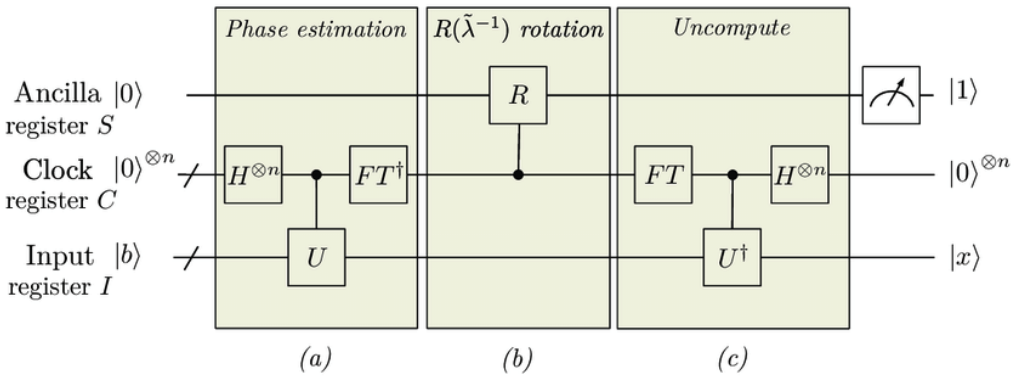

Figura 3.3: Schema del circuito dell'algoritmo HHL[10]

3.3.1 Preparazione dello stato iniziale

Per ipotesi la matrice A è Hermitiana e quindi normale, dunque per il teorema spettrale in campo complesso, sappiamo che è possibile esprimere A come $A = \sum_j \lambda_j |u_j\rangle \langle u_j|$, dove λ_j e $|u_j\rangle$ sono rispettivamente autovalori e autovettori della matrice, inoltre i vettori $|u_j\rangle$ risultano essere ortogonali dunque $\langle u_i|u_j\rangle = \delta_{ij}$.

Teorema spettrale complesso

Sia $\langle , \rangle$ il prodotto Hermitiano standard su $\mathbb{C}^n$, e sia A una matrice normale $n \times n$. Allora, esiste una base ortonormale di $\mathbb{C}^n$ costituita da autovettori di A.

La dimostrazione viene esposta in appendice A.

Definizione di prodotto hermitiano

Sia $\mathbb{V}$ uno spazio vettoriale definito su $\mathbb{C}$, n-dimensionale. Si dice prodotto hermitiano su $\mathbb{V}$ un'applicazione

$$\langle , \rangle : \mathbb{V} \times \mathbb{V} \to \mathbb{C}$$
$$(v, w) \mapsto \langle v, w \rangle \in \mathbb{C}$$

che gode delle seguenti proprietà:

a) Linearità rispetto al primo termine:

$$\forall v_1, v_2, w \in \mathbb{V} \quad \text{e} \quad \forall \lambda \in \mathbb{C}$$
$$\langle v_1 + v_2, w \rangle = \langle v_1, w \rangle + \langle v_2, w \rangle$$
$$\langle \lambda v_1, w \rangle = \lambda \langle v_1, w \rangle$$

b) Simmetria:

$$\langle w, v \rangle = \overline{\langle v, w \rangle} \quad \forall v, w \in \mathbb{V}$$

dove $\overline{\langle v, w \rangle}$ indica il complesso coniugato di $\langle v, w \rangle$

c) Antilinearità rispetto al secondo termine:

$$\forall v, w_1, w_2 \in \mathbb{V} \quad \text{e} \quad \forall \lambda \in \mathbb{C}$$
$$\langle v, w_1 + w_2 \rangle = \langle v, w_1 \rangle + \langle v, w_2 \rangle$$
$$\langle v, \lambda w_1 \rangle = \overline{\lambda} \langle v, w_1 \rangle$$

Tornando all'algoritmo, data la matrice $A = \sum_j \lambda_j |u_j\rangle \langle u_j|$, consideriamo il caso in cui il vettore $|b\rangle$ corrisponde all'autovettore $|u_j\rangle$ della matrice A, questa ipotesi verrà rilassata in seguito. Sia tale vettore rappresentato nel circuito da un numero di qubit sufficiente a rappresentarlo con la precisione scelta, i qubit utilizzati compongono l'*input register*, indicato con la lettera I nella figura 3.3. Essendo A Hermitiana, la matrice $U = e^{iA}$ è unitaria con autovalori della forma $e^{i\lambda_j} = e^{2\pi i \phi}$, dove le λ_i sono gli autovalori di A. Dunque per trovare gli autovalori di A possiamo trovare la fase ϕ degli autovalori di U. Per questo

motivo abbiamo bisogno di un registro di qubit che contenga le fasi risultanti dalla Phase Estimation, che equivalgono agli autovalori della matrice A. Il numero n di qubit che compongono il *clock register*, indicato in figura con C, è dato dalla relazione presentata nella sezione 3.1.2. Tali qubit vengono posti nello stato $|0\rangle$.

3.3.2 Phase Estimation

Per quanto visto nella sezione 3.1.2, possiamo utilizzare la *QPE* per stimare le fasi (parte (a) della figura 3.3). Partendo dallo stato $|0\rangle^{\otimes n}|u_j\rangle$ otteniamo lo stato $|\tilde{\lambda_j}\rangle|u_j\rangle$, $\tilde{\lambda_j}$ è la rappresentazione binaria in n bit di λ_j, autovalore di $|u_j\rangle$.

3.3.3 Rotazioni controllate

L'operatore rotazione è definito da

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

Esso agisce su un qubit, che si trova inizialmente nello stato $|0\rangle$, detto *ancilla* qubit, indicato con S nella figura 3.3, trasformandolo nello stato $\cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$. Tali operatori rotazione hanno come qubit di controllo quelli contenuti nel *clock register* (parte (b) della figura 3.3). 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 $|\tilde{\lambda_j}\rangle$.

Sia ora uno stato di input $|b\rangle$ qualsiasi, esso può essere espresso come $|b\rangle = \sum_j \beta_j |u_j\rangle$, dunque le rotazioni controllate generano lo stato

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

Infatti abbiamo inizialmente uno stato formato da $|b\rangle = \sum_j \beta_j |u_j\rangle$, gli n qubit nello stato $|0\rangle$ del *clock register* e il qubit *ancilla* nello stato $|0\rangle$. Dunque il nostro stato di partenza è

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

dopo la *QPE* esso diventa

$$|0\rangle \sum_j \beta_j |\tilde{\lambda_j}\rangle |u_j\rangle$$

infine dopo aver applicato le rotazioni controllate otteniamo

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

Scegliendo l'angolo delle rotazioni controllate $\theta = \arcsin\frac{C}{\tilde{\lambda}}$ con C costante di normalizzazione, otteniamo lo stato

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

I calcoli appena mostrati rappresentano la versione più generica dell'algoritmo in cui la valutazione dei reciproci degli autovalori viene inglobata nelle rotazioni controllate. Tale valutazione può anche essere effettuata *costruendola* nel circuito, cioè applicando degli operatori che permettano di svolgere

$$|0\rangle \sum_j \beta_j |\tilde{\lambda_j}\rangle |u_j\rangle \quad \rightarrow \quad |0\rangle \sum_j \beta_j |\frac{1}{\tilde{\lambda_j}}\rangle |u_j\rangle$$

senza misurare i qubit. In questo modo cambierà la dipendenza tra le rotazioni controllate e i qubit. Tale metodo può essere utile sia didatticamente, sia in casi molto particolari. In questa tesi si è scelto di costruire in modo semplice un sistema per valutare i reciproci $\frac{1}{\lambda_j}$ degli autovalori prima di effettuare le rotazioni controllate. In generale tale operazione viene effettuata sapendo che l'inversa della matrice A può essere espressa come $A^{-1} = \sum_j \frac{1}{\lambda_j} |u_j\rangle \langle u_j|$. Il calcolo risulta essere una procedura non banale, anche l'articolo originale [10] non lo tratta. In questa tesi verrà utilizzata una matrice A per cui il reciproco degli autovalori può essere valutato in modo semplice. Nell'articolo [11] si trova il circuito generale per l'inversione degli autovalori di una matrice qualsiasi A.

3.3.4 Uncomputation

Applicando le procedure inverse dell'inversione degli autovalori e della *QPE* (parte (c) della figura 3.3) riportiamo il *clock register* nello stato iniziale $|0\rangle^{\otimes n}$. Otteniamo quindi lo

stato

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

3.3.5 Misure

Misurando il qubit *ancilla* e il registro di input e considerando valide solo le misure in cui il qubit *ancilla* si trova nello stato $|1\rangle$, otteniamo nel registro di input uno stato con ampiezze proporzionali al vettore classico delle soluzioni. Infatti dalla formula dello stato finale ottenuto è possibile notare che se il qubit *ancilla* si trova nello stato $|1\rangle$, allora il registro I si trova nello stato $|\hat{x}\rangle = A^{-1}|b\rangle$ dove $\frac{|\hat{x}\rangle}{\|\hat{x}\|} = |x\rangle$ soluzione del problema. L'algoritmo è costruito in modo tale che misurando e quindi proiettando lo stato finale, venga persa l'informazione non utile alla risoluzione del problema, si ricordi infatti il concetto di misura spiegato nella sezione 1.3.

3.4 Vantaggio teorico e limitazioni

Come riportato in precedenza, l'algoritmo HHL porta uno speedup esponenziale rispetto al miglior algoritmo classico per la risoluzione di sistemi lineari. Infatti il miglior algoritmo classico va in un tempo $\propto O(N \log(\frac{1}{\varepsilon}))$ con ε che rappresenta la precisione del risultato, invece l'algoritmo HHL va come $O(\frac{\log N}{\varepsilon})$.

Il problema di molti algoritmi quantistici sta però nella lettura dei risultati, infatti abbiamo visto che l'HHL incorpora la soluzione nel registro di input, che non può essere letto come fosse composto da bit. La soluzione è codificata nelle ampiezze dello stato quantistico in sovrapposizione , dunque se andassimo a misurare il vettore *soluzione* faremmo collassare lo stato, come visto nella sezione 1.3, trovando uno degli stati possibili ma di fatto perdendo l'informazione cercata.

Una possibilità per la lettura diretta complessiva del risultato è l'iterazione di esecuzioni del programma per un numero considerevole di volte, tale procedura richiederebbe però un tempo $\sim O(n)$ vanificando l'accelerazione ottenuta con l'algoritmo quantistico.

Spesso però non è necessario leggere direttamente la soluzione complessiva, infatti nella

maggior parte dei casi l'algoritmo HHL può essere utilizzato come subroutine di algoritmi in cui sono necessari solo dei *campionamenti* del vettore soluzione, o addirittura il vettore soluzione $|x\rangle$ non deve neanche essere misurato perchè rappresenta uno stato intermedio di una computazione maggiore.

3.5 Test dell'algoritmo

3.5.1 IBM qunatum experience e Qiskit

La realizzazione pratica dell'algoritmo HHL è stata possibile grazie all'IBM quantum experience, servizio gratuito che IBM mette a disposizione di chiunque si crei un account. In tale piattaforma è possibile accedere da remoto a processori quantistici reali, basati su qubit a superconduttori. I sistemi disponibili gratutitamente hanno un numero di qubit limitato ma sufficiente allo scopo di questa tesi. I processori messi a disposizione da IBM sono di tipo NISQ, dunque soggetti a rumore quantistico. Ci si aspetta quindi che i risultati della computazione non abbiano un grado di precisione ottimale. Per implementare l'algoritmo si è utilizzato Qiskit, un software kit scientifico open source per la computazione quantistica realizzato da IBM. Qiskit, scritto in Python, permette tramite un set di comandi, la realizzazione e la manipolazione di circuiti quantistici. Grazie a Qiskit è anche possibile eseguire i propri programmi su simulatori classici. Una procedura molto importante effettuata automaticamente da questo software kit è il *transpiling* che consiste nell'ottimizzazione dei circuiti dati in input attraverso la semplificazione dei gate. Tale procedura è necessaria perchè l'utente ha a disposizione un numero non indifferente di porte logiche ma il 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 quindi di tradurre i gate definiti dall'utente in gate eseguibili sul processore quantistico. Oltre a questo, il software kit prodotto da IBM, offre tre backend locali di simulazione di circuiti quantistici:

- Qasm Simulator: esegue il circuito un numero definito di volte (multi-shot execution) e restituisce il conteggio dei risultati ottenuti dalla misura dei qubit

- Statevector Simulator: esegue il circuito una volta sola (single-shot) e permette di ottenere la funzione d'onda dello stato finale rappresentato dai qubit

- Unitary Simulator: esegue il circuito simulando un computer quantistico non prono agli errori (fault tollerant) e restituisce l'espressione matriciale dell'operatore unitario a cui il circuito è equivalente

3.5.2 Premesse

In questa tesi l'algoritmo HHL viene utilizzato per risolvere un sistema formato da una matrice 2×2 particolare. Tale scelta è dovuta al fatto che il circuito può raggiungere un alto grado di complessità per matrici qualsiasi, in particolare vedremo che la matrice scelta permette di valutare i reciproci degli autovalori in modo semplice. Verranno presentati i risultati ottenuti implementando l'algoritmo con 4 qubit, 2 dei quali verranno utilizzati per la *QPE*.

Valutazione dei reciproci degli autovalori

Come accennato in precedenza la matrice A scelta per effettuare i test, risulta essere una matrice particolarmente favorevole, infatti scelta

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

i suoi autovalori sono $\lambda_1 = 1$ e $\lambda_2 = 2$, con autovettori corrispondenti rispettivamente $|u_1\rangle = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ e $|u_2\rangle = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Valutare i reciproci di tali autovalori risulta essere una procedura molto semplice con un preciso accorgimento, infatti basta utilizzare uno *SWAP* gate tra il qubit $q1$ e $q2$. Per chiarire mostriamo (i qubit interessati dalla *QPE* sono i 2 due centrali):

$$1 = 0001_{(2)} \xrightarrow{SWAP_{1-2}} 0001_{(2)} = 1$$
$$2 = 0010_{(2)} \xrightarrow{SWAP_{1-2}} 0100_{(2)} = \frac{1}{2}$$

in questo modo è quindi possibile ottenere i reciproci degli autovalori in modo semplice, tale passaggio, come accennato in precedenza, richiede una computazione difficoltosa nel caso generale.

Processione dei risultati

Il vettore di input $|b\rangle$ utilizzato per i test è $|b\rangle = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.4472 \\ 0.8944 \end{bmatrix}$, ci si aspetta quindi dalla computazione lo stato-soluzione normalizzato in output $|x\rangle$:

$$A^{-1}|b\rangle = \frac{|\hat{x}\rangle}{\|\hat{x}\|} = |x\rangle$$

$$|x\rangle = \frac{1}{\sqrt{26}} \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 0.196 \\ 0.981 \end{bmatrix}$$

Nel test con 4 qubit lo Statevector simulator fornisce uno stato finale che ha coefficienti uguali a zero per uno stato qualunque del clock register che non sia $|00\rangle$. Ciò significa che l'implementazione risulta concorde con la teoria.
I test forniscono vettori soluzione della forma:

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

Il primo qubit è l'input register, il secondo è l'*ancilla* qubit. Come spiegato nella sezione 3.3.5, i coefficienti utili sono solo β_{01} e β_{11}, cioè i coefficienti degli stati in cui l'*ancilla* qubit si trova nello stato $|1\rangle$. In generale non è possibile accedere a tale stato, come si vedrà nel test sul computer quantistico reale. Dai test si otterranno in realtà le probabilità sulla base computazionale, nel test effettuato sul simulatore *qasm* sarà possibile, per motivi che verranno spiegati in seguito, ricostruire lo stato e dunque il vettore soluzione. I coefficienti utili dovranno essere normalizzati per ricostruire lo stato soluzione. Il procedimento di normalizzazione consiste in

$$\tilde{\beta_{01}} = \frac{\beta_{01}}{\sqrt{\beta_{01}^2 + \beta_{11}^2}} \qquad \tilde{\beta_{11}} = \frac{\beta_{11}}{\sqrt{\beta_{01}^2 + \beta_{11}^2}}$$

dunque la soluzione sarà data da

$$|\tilde{x}\rangle = \tilde{\beta_{01}}|01\rangle + \tilde{\beta_{11}}|11\rangle$$

3.5.3 Test

In questo test sono stati utilizzati 4 qubit totali, 2 per la *QPE*. Il numero di qubit usati per la *QPE* equivale al numero di bit usati per rappresentare gli autovalori. Dato che la procedura di *transpile* di Qiskit non supporta gli esponenziali di matrice presenti nella subroutine per stimare le fasi, si è utilizzata un'implementazione *ad hoc* per la matrice *A*, dallo spunto del circuito in [12].

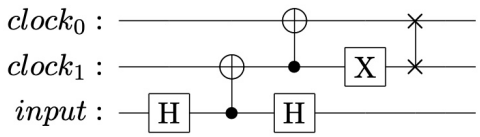

Figura 3.4: Circuito *ad hoc* della *QPE*

Con tale subroutine si è costruito l'intero circuito che implementa l'algoritmo HHL:

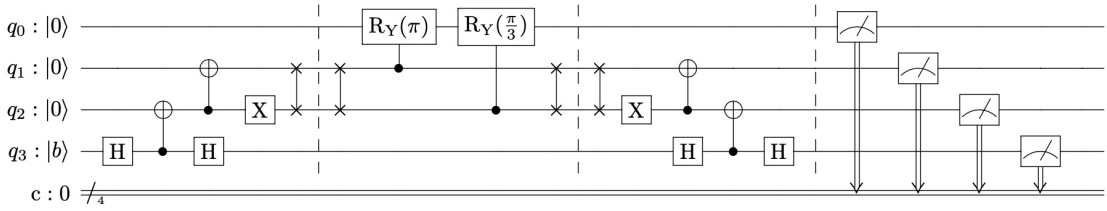

Figura 3.5: Circuito *ad hoc* dell'algoritmo HHL

Una volta fatto girare l'algoritmo, utilizzando come qubit di input *b*, si sono ottenuti i risultati

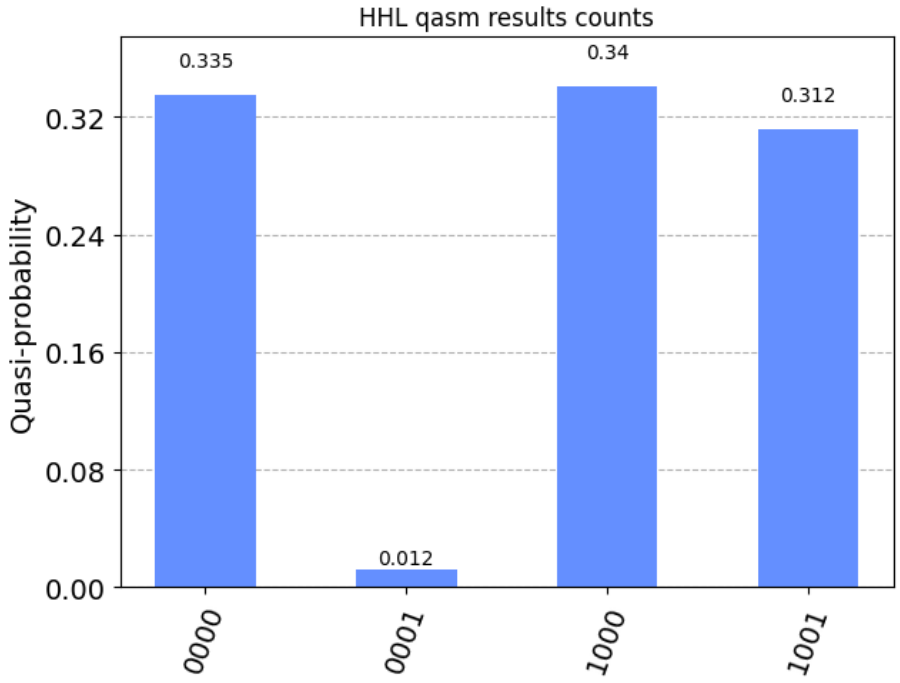

Figura 3.6: Probabilità degli stati del test (qasm simulator)

Il grafico fornisce le probabilità sulla base computazionale, cioè i moduli quadri delle ampiezze degli stati della base computazionale. In generale non si ha accesso alle ampiezze ma solo alle probabilità, dunque non si è in grado di ricostruire lo stato soluzione. Per questo motivo per verificare la correttezza dello stato soluzione, a cui non si ha accesso diretto, si effettua un test con le probabilità. In questo elaborato si è considerato un sistema 2×2 e si cerca la soluzione normalizzata, dunque è possibile effettuare il test di correttezza confrontando i rapporti tra i moduli quadri della soluzione corretta e i rapporti tra i moduli quadri della soluzione ottenuta dall'algoritmo. In tal senso si verifica la correttezza della computazione senza *leggere* la soluzione ottenuta.

In fig. 3.6 il primo valore di ogni stato rappresenta il qubit di input e l'ultimo il qubit *ancilla*, dunque le probabilità da tenere in considerazione sono

$$\gamma_{01} = 0.012 \quad \text{e} \quad \gamma_{11} = 0.312$$

I rapporti tra i moduli quadri delle entrate della soluzione teorica e della soluzione ottenuta sono

$$rapporto_{esatta} = \frac{0.196^2}{0.981^2} = 0.0399 \qquad rapporto_{trovata} = \frac{0.012}{0.312} = 0.0384$$

L'accuratezza della computazione può essere esplicitata dal rapporto $\frac{rapporto_{esatta}}{rapporto_{trovata}} = 0.9635$ che mostra un'alta precisione della soluzione trovata.

Il circuito è stato costruito in modo che le ampiezze siano reali e positive, possiamo quindi considerare le radici dei moduli quadri come le ampiezze sulla base computazionale e ricostruire lo stato con i coefficienti

$$\beta_{00} = \sqrt{0.335} \quad \beta_{01} = \sqrt{0.012} \quad \beta_{10} = \sqrt{0.34} \quad \beta_{11} = \sqrt{0.312}$$

ottenendo lo stato

$$|\psi_{qasm}\rangle = \sqrt{0.335}\,|00\rangle + \sqrt{0.012}\,|01\rangle + \sqrt{0.34}\,|10\rangle + \sqrt{0.312}\,|11\rangle$$

Per quanto visto in precedenza, i coefficienti da tenere in considerazione sono

$$\beta_{01} = \sqrt{0.012} \qquad \beta_{11} = \sqrt{0.312}$$

Normalizzandoli si ottiene:

$$\tilde{\beta_{01}} = \frac{\sqrt{0.012}}{\sqrt{0.012+0.312}} = 0.1924 \quad \tilde{\beta_{11}} = \frac{\sqrt{0.312}}{\sqrt{0.012+0.312}} = 0.9813$$

Quindi si è ottenuto lo stato normalizzato

$$|\tilde{x}_{qasm}\rangle = \begin{bmatrix} 0.1924 \\ 0.9813 \end{bmatrix}$$

Oltre al rapporto visto in precedenza, sapendo a priori che lo stato corretto non ha fasi relative e che il simulatore *qasm* è ideale, è possibile calcolare la fidelity come $|\,\langle \tilde{x}_{qasm}|x\rangle\,|^2$, ottenendo 1.00. Confermando quindi che l'algoritmo testato ha un'ottima prestazione nella risoluzione del sistema lineare preso in esame.

Si è ripetuto lo stesso esperimento ma utilizzando un computer quantistico reale di IBM e sono stati ottenuti i risultati

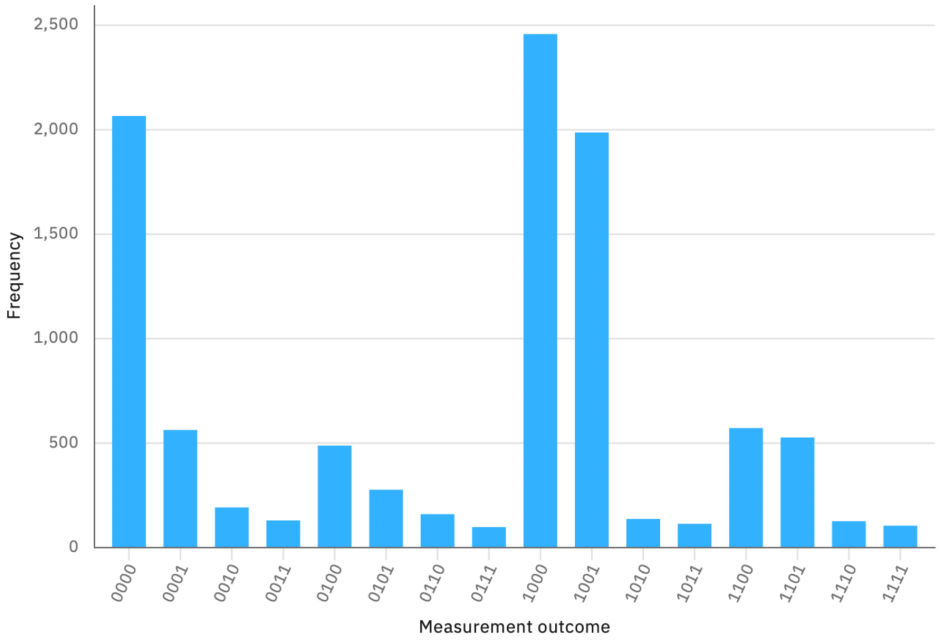

Figura 3.7: Probabilità degli stati del test (quantum computer ibm_perth)

Si nota subito che la computazione effettuata da un computer quantistico reale risulta essere meno precisa, infatti, secondo la teoria, al termine dell'algoritmo lo stato finale avrà i qubit del *Clock* register nello stato $|00\rangle$, invece dalla figura 3.7, si nota che anche gli stati con diverso stato del *Clock* register hanno probabilità non nulla. In questo caso non è possibile ricavare le ampiezze dalle probabilità, infatti a causa del rumore quantistico potrebbero essersi create fasi relative, non previste dal calcolo esatto teorico, che non si è in grado di vedere dal grafico 3.7, dato che i dati ottenuti sono i moduli quadri delle ampiezze. A differenza del caso precedente, non è possibile ricostruire lo stato finale e dunque ottenere esplicitamente la soluzione trovata. Come spiegato nel paragrafo 3.4, una limitazione dell'algoritmo HHL, come di molti altri algoritmi quantistici, consiste nell'impossibilità di *leggere* la soluzione finale, essendo essa codificata nelle ampiezze di uno stato quantistico. Per avere il vettore completo si dovrebbe effettuare la tomografia, perdendo il vantaggio computazionale dell'algoritmo. Per valutare la correttezza del risultato è possibile effettuare il test del rapporto come fatto nel caso del simulatore, in questo modo è possibile essere a conoscenza del grado di accuratezza del vettore soluzione ottenuto,

senza avere diretto accesso alle sue entrate.

Le probabilità da tenere in considerazione sono

$$\gamma_{01} = 0.0563 \quad \text{e} \quad \gamma_{11} = 0.1987$$

I rapporti tra i moduli quadri delle entrate della soluzione teorica e della soluzione ottenuta sono

$$rapporto_{esatta} = \frac{0.196^2}{0.981^2} = 0.0399 \qquad rapporto_{trovata} = \frac{0.0563}{0.1987} = 0.2833$$

Risulta quindi che il rumore quantistico a cui è soggetto il computer quantistico *perth* porta a risultati fortemente influenzati da errore. Considerando lo stato attuale dell'hardware quantistico e che i computer quantistici messi a disposizione da IBM sono soggetti a numerose parametrizzazioni che possono influenzare positivamente ma anche negativamente alcune computazioni, il risultato ottenuto è soddisfacente ma non preciso come quello ottenuto con il simulatore qasm. Si noti comunque che l'istogramma mantiene la struttura sperata in cui le probabilità dello stato $|0000\rangle$ e $|1000\rangle$ assumono più o meno gli stessi valori.

Conclusioni

In questa tesi sono stati presentati e testati due algoritmi fondamentalmente diversi, il metodo del gradiente coniugato sfrutta la computazione classica per risolvere un sistema lineare, invece l'algoritmo HHL sfrutta la computazione quantistica. Si sono presentate le principali differenze tra le due e come un approccio quantistico può risultare esponenzialmente più veloce, almeno a livello teorico. I test effettuati con il metodo del gradiente coniugato hanno portato ad ottimi risultati quando applicato a matrici ben condizionate e il numero di iterate necessarie al raggiungimento della soluzione sono in linea con la teoria. Al contrario per matrici mal condizionate si è rivelato non altrettanto efficace, arrestandosi, per via del test d'arresto, prima di convergere alla soluzione desiderata ma superando il numero di iterazioni previste dalla teoria a causa della propagazione degli errori. L'algoritmo HHL è stato testato su una matrice particolare per permettere una maggior semplicità di implementazione, i risultati ottenuti sono in linea con le aspettative, considerando il livello attuale degli hardware quantistici e non avendo applicato tecniche di *error correction*. I risultati ottenuti con il simulatore *qasm* hanno una fidelity molto alta, infatti il rapporto tra le probabilità della soluzione trovata è 0.0384 e quello della soluzione esatta è 0.0399. I test sul computer quantistico reale *perth* non hanno portato ad una soluzione altrettanto buona, ciò è causato dal rumore quantistico che provoca errori nella computazione. Il grafico delle probabilità presenta la stessa struttura vista nel test *ideale* dimostrando che l'algoritmo è stato eseguito correttamente, dunque l'errore nella soluzione è dovuto unicamente al rumore quantistico dell'hardware. Il caso di matrici mal condizionate non è stato considerato ma l'approccio risulta essere alquanto particolare, infatti come spiegato nella pubblicazione originale [10], un modo per gestire il problema del numero di condizionamento è dividere la matrice in due parti ottenendo una matrice ben condizionata e una mal condizionata e *ignorare* nella fasi di inversione la parte mal condizionata.

A. Teorema spettrale complesso

Per dimostrare il teorema spettrale complesso avremo bisogno di un lemma:

Lemma 1

Siano A e B due matrici complesse $n \times n$ tali che $AB = BA$. Allora esiste un elemento non nullo di $\mathbb{C}^n$ che è autovettore sia di A, sia di B.

Teorema spettrale complesso

Sia $\langle , \rangle$ il prodotto Hermitiano standard su $\mathbb{C}^n$, e sia A una matrice normale $n \times n$. Allora, esiste una base ortonormale di $\mathbb{C}^n$ costituita da autovettori di A.

Dimostrazione. Procediamo per induzione sulla dimensione n. Il teorema è ovvio per $n = 1$ (A è una matrice 1×1). Dobbiamo dimostrare il teorema spettrale in dimensione n, supponendo che esso sia vero in dimensione $n-1$. Poichè A commuta con A^*, A e A^* hanno almeno un autovettore in comune per il lemma 1: indichiamolo con v_1. Dividendolo per la sua norma, potremo supporre che v_1 abbia lunghezza 1. Ora, siano $\mathbb{V}_1 = \text{Span}(v_1)$ e $\mathbb{W} = (\mathbb{V}_1)^\perp$. A manda $\mathbb{W}$ in sè stesso: se $w \in \mathbb{W}$,

$$\langle Aw; v_1 \rangle = \langle w; A^* v_1 \rangle = \mu \langle w; v_1 \rangle = 0$$

La penultima uguaglianza si ottiene ricordando che v_1 è un autovettore anche per A (indicando con μ l'autovalore associato). L'ultima uguaglianza si ottiene ricordando che $w \in \mathbb{W} = (\mathbb{V}_1)^\perp$. Dunque, Aw è ortogonale a v_1, e cioè $Aw \in \mathbb{W}$. A può essere vista come un'applicazione lineare di $\mathbb{W}$ in sè stesso. Ma $\mathbb{W}$ ha dimensione $n-1$, dunque si può applicare l'ipotesi induttiva. Sia $\{v_2; \ldots; v_n\}$ la base ortonormale di $\mathbb{W}$. Allora, essendo v_1 ortogonale ad ogni vettore di $\mathbb{W}$, $\{v_1; v_2; \ldots; v_n\}$ è la base cercata di $\mathbb{C}^n$.

□

Il teorema e la dimostrazione sono stati presi dagli appunti del Professore Roberto Catenacci [13].

Bibliografia

[1] Preskill, J. «Quantum Computing in the NISQ era and beyond». In: *Quantum* 2 (ago. 2018), p. 79. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331%2Fq-2018-08-06-79 (cit. a p. 1).

[2] Shor, P. W. «Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer». In: *SIAM Journal on Computing* 26.5 (ott. 1997), pp. 1484–1509. DOI: 10.1137/s0097539795293172. URL: https://doi.org/10.1137%2Fs0097539795293172 (cit. a p. 1).

[3] Grover, L. K. *A fast quantum mechanical algorithm for database search*. 1996. arXiv: quant-ph/9605043 [quant-ph] (cit. a p. 1).

[4] Feynman, R. P. «Simulating physics with computers». In: *International Journal of Theoretical Physics* 21.6 (1982), pp. 467–488. DOI: 10.1007/BF02650179. URL: https://doi.org/10.1007/BF02650179 (cit. a p. 2).

[5] Dirac, P. A. M. «A new notation for quantum mechanics». In: *Mathematical Proceedings of the Cambridge Philosophical Society* 35.3 (1939), pp. 416–418. DOI: 10.1017/S0305004100021162 (cit. a p. 3).

[6] Nielsen, M. A. e Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667 (cit. alle pp. 3, 6, 8).

[7] Hidary, J. *Quantum Computing: An Applied Approach*. Springer International Publishing, 2021. ISBN: 9783030832742. URL: https://books.google.it/books?id=T7VFEAAAQBAJ (cit. a p. 7).

[8] Shewchuk, J. R. «An Introduction to the Conjugate Gradient Method Without the Agonizing Pain». In: 1994. URL: https://api.semanticscholar.org/CorpusID:6491967 (cit. a p. 16).

[9] Nocedal, J. e Wright, S. J. *Numerical Optimization*. 2e. New York, NY, USA: Springer, 2006 (cit. a p. 20).

[10] Harrow, A. W., Hassidim, A. e Lloyd, S. «Quantum Algorithm for Linear Systems of Equations». In: *Phys. Rev. Lett.* 103 (15 ott. 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502 (cit. alle pp. 26, 35, 38, 48).

[11] Cao, Y. et al. «Quantum circuit design for solving linear systems of equations». In: *Molecular Physics* 110.15-16 (ago. 2012), pp. 1675–1680. DOI: 10.1080/00268976.2012.668289. URL: https://doi.org/10.1080%2F00268976.2012.668289 (cit. a p. 38).

[12] Cai, X.-D. et al. «Experimental Quantum Computing to Solve Systems of Linear Equations». In: *Physical Review Letters* 110.23 (giu. 2013). DOI: 10.1103/physrevlett.110.230501. URL: https://doi.org/10.1103%2Fphysrevlett.110.230501 (cit. a p. 43).

[13] Catenacci, R. «Appunti su alcune classi importanti di matrici e gruppi di matrici». In: (2013). URL: https://people.unipmn.it/catenacc/geo/matrici.pdf (cit. a p. 49).