POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Matematica

Tesi di Laurea Magistrale

Quantum Computing e Risk Analysis: studio di un caso reale

Supervisori:

Prof. Barbara Trivellato

Ing. Giuseppe Cotugno

Candidato:

Edoardo Chiatello

Anno Accademico 2019 - 2020

Sommario

L'obiettivo che si pone questa tesi è l'analisi di un caso specifico di simulazione stocastica applicata a Risk Analysis tramite l'utilizzo delle tecnologie quantistiche. Nella fattispecie, si delineano gli algoritmi che possono essere applicati a Quantum Computing, dei quali si fornisce la codifica per l'esecuzione su quantum computer reali. Gli strumenti che vengono sfruttati comprendono i server, i development environment e tutte le risorse messe a disposizione da IBM all'interno delle piattaforme open-source. L'esecuzione su quantum computer mira a comparare i risultati ottenuti dagli elaboratori quantistici e quelli classici; valutare il corretto bilanciamento tra profondità dei circuiti di calcolo quantistici e la gestione delle condizioni di errore, in relazione alle capacità di calcolo degli attuali quantum computer. Lo studio porta, di fatto, ad una sintesi sullo stato dell'arte del Quantum Computing e le sue potenzialità.

Indice

Elenco delle tabelle

Elenco delle figure

Acronimi

ASRF

Asymptotic single risk factor

BSM

Black-Scholes-Merton

CDF

Cumulative Distribution Function

CVaR

Conditional Value at Risk

EAD

Exposure At Default

ES

Expected Shortfall

IQAE

Iterative Quantum Amplitude Estimation

IRB

Internal-Rated Based

LGD

Loss Given Default

MC

Monte Carlo

MLAE

Maximum Likelihood Amplitude Estimation

NISQ

Noisy Intermediate-Scale Quantum

PDF

Probability Density Function

QAA

Quantum Amplitude Amplification

QAE

Quantum Amplitude Estimation

QC

Quantum Computing

QFT

Quantum Fourier Transformation

VaR

Value at Risk

WHP

With High Probability

Capitolo 1

Introduzione

Il Quantum Computing è un dominio scientifico che si fonda su nozioni di Computer Science, Matematica e Fisica. Esso costituisce un nuovo metodo computazionale che sfrutta un insieme di meccanismi, i quali basano il proprio funzionamento sulle leggi della meccanica quantistica.
La teoria a cui si fa riferimento comprende un insieme di fenomeni che sono stati scoperti ed indagati già nella prima metà del secolo scorso. Un esempio tra tutti è la condizione di *entaglement*, che mette in relazione due o più particelle fisiche di uno stesso insieme meccanico. I quantum computer furono teorizzati in seguito alla scoperta di questi fenomeni. Tuttavia, l'ipotesi di avanguardia, che aveva proposto la possibilità di poter creare macchine quantistiche, non ricevette immediato sostegno da parte delle tecniche di costruzione di tali oggetti. Infatti, solamente a partire dallo scorso decennio, le tecnologie a disposizione della comunità scientifica hanno concesso la nascita dei primi esemplari di quantum computer. Inoltre, ciò è stato possibile solo grazie all'intervento delle grandi multinazionali High-Tech, che hanno deciso di investire ingenti capitali in questo ambito di ricerca. Un primo esempio è IBM, che ha progettato un hardware quantistico che prende il nome di *IBM Q System One*. L'azienda stantunitense crede fermamente nelle potenzialità del Quantum Computing e, per agevolarne lo sviluppo, ha deciso di progettare una piattaforma online per mettere a disposizione di chiunque lo desideri le conoscenze ed i risultati raggiunti nelle ricerche interne. Inoltre, grazie

Figura 1.1: La calotta fisica del IBM Q System One ha lo scopo di conservare le temperature estreme di raffreddamento dei materiali superconduttori che costituiscono i circuiti interni.

a questo strumento, è consentito il libero accesso ai reali hardwares quantistici di IBM. In altro modo, infatti, attualmente non sarebbe possibile per nessun privato studioso poter possedere un quantum computer personale. Oltre al costo economico esorbitante, bisogna pensare che esso è costituito fisicamente da una macchina di dimensioni paragonabili a quelle dei primi computer classici che furono sviluppati nello scorso decennio (Figura 1.1). Inoltre, esso esige un'enorme quantità di energia per poter mantenere le temperature estremamente basse all'interno di alcune sezioni costituenti.

Pertanto, si può sostenere che la teoria quantistica, insieme agli algoritmi che sfrutterebbero le sue potenzialità, si trova in netto vantaggio rispetto ad un suo riscontro pratico. Solamente con la nascita degli hardwares reali risulta possibile intraprendere la fase sperimentale per ottenere le necessarie evidenze empiriche a favore degli algoritmi citati. In questo momento i quantum computer non sono ancora in grado di poter simulare sistemi fisici complessi, in quanto la loro fattura è recente. Per di più, essi sono affetti da errori di *decoerenza*, dovuta alla difficoltà di isolamento del sistema meccanico dall'esterno. Quindi, oltre ad essere ancora in

una fase di sviluppo iniziale che consente di trattare problemi di bassa complessità, essi producono dei risultati che sono affetti da distorsioni di notevole entità.
Le potenzialità che caratterizzano il Quantum Computing hanno permesso ai ricercatori di credere che esso non sostituirà la computazione classica in tutti i suoi utilizzi. Piuttosto, esso fornirà una valido apporto pratico per la risoluzione di problemi che sono difficilmente trattabili, o addirittura impossibili, per un elaboratore classico. La causa di ciò risiede nel fatto che alcuni problemi matematici richiedono l'esecuzione di un numero estremo di istruzioni algoritmiche, che portano all'abbandono di trovare una soluzione deterministica con metodi di computazione classica. Di contro, i computer quantistici possono essere in grado di fornire un output corretto per problemi di questo tenore in tempi decisamente più brevi.
A favore di questa tesi, basti pensare che in un elaboratore quantistico costituito da N unità di informazione, i cosiddetti *qubits*, permettono la rappresentazione di 2^N stati differenti. Il medesimo numero può essere replicato su un computer classico tramite l'utilizzo di 2^N bits. Risulta immediato verificare che questa situazione verte maggiormente a favore del primo caso quando il numero di unità informative richieste aumenta.

Svariati sono gli ambiti in cui il Quantum Computing può trovare applicazione. I principali riguardano la simulazione di sistemi chimici, ma allo stesso tempo trova interessanti risvolti in ambito di Ottimizzazione e Data Science. Esistono infatti alcune tecniche quantistiche che sono sostanzialmente delle repliche dei metodi noti di Machine Learning, come ad esempio Principal Component Analysis.
Tuttavia, il contesto di indagine che intende affrontare questa tesi è finanziario. Nella fattispecie, verrà presentato un caso di studio che rientra tra quelli distribuiti in Qiskit, che è la piattaforma open-source di IBM [1]. Esso riguarda l'ambito di Credit Risk Analysis e del calcolo delle più comuni misure di rischio sulla distribuzione di perdita di un portafoglio azionario. Una di queste è il *Value at Risk*, che ricopre il ruolo centrale di monitoraggio per la gestione di controllo di liquidità di un portafoglio. Nel caso specifico, si parla di un portafoglio appartenente ad un istituto finanziario. Indi per cui, la distribuzione di default viene stimata tramite un modello *ASRF* che è previsto dagli accordi internazionali di Basilea. Come

verrà mostrato nel seguito, il metodo con cui viene calcolato il VaR sfrutta l'utilizzo di un determinato algoritmo quantistico che prende il nome di *Quantum Amplitude Estimation*. Esso funge da leva per poter asserire che il Quantum Computing può costituire uno strumento di calcolo più performante di quelli conosciuti di stampo classico. Infatti, a parità di incertezza sul risultato, QAE produce un output con un numero di ordine quadratico inferiore rispetto ai metodi classici Monte Carlo. Si parla infatti di *Quantum speed-up*. Tuttavia questa tesi è stata avanzata tramite una dimostrazione teorica. Solo con l'avvento di un hardware capace di simulare i problemi reali, si potrà avere un riscontro pratico che contribuirà alla sua conferma oppure alla temporanea invalidazione.

Di seguito vengono elencati i capitoli in cui si snoda l'argomentazione della tesi. Per ognuno vengono elencati i temi trattati:

- Capitolo 2: vengono affrontati nello specifico i fondamenti della meccanica quantistica che permettono il sorgere della computazione quantistica. Dopodiché si forniscono i metodi di gestione di quest'ultima e come avviene la codifica dell'informazione reale all'interno di un sistema quantistico;

- Capitolo 3: annovera i principali algoritmi di stampo quantistico, come ad esempio il *Grover's Algorithm*. Questo capitolo segue il processo di sviluppo che, a partire dagli algoritmi più semplici, porta alla definizione del Quantum Amplitude Estimation;

- Capitolo 4: spiega il contesto finanziario di applicazione. Ma soprattutto fornisce una trattazione analitica del modello ASRF che viene sfruttato per gestire la distribuzione di perdita del portafoglio azionario;

- Capitolo 5: viene definito il circuito quantistico che è implementato tramite istruzioni Python del package Qiskit. Segue il metodo presentato in [2]. Inoltre analizza la codifica che mappa l'output del circuito con le stime dei risultati;

- Capitolo 6: viene preso in esame un esempio numerico ed eseguito su hardware reale. I risultati ottenuti sono comparati con quelli derivanti da una simulazione

Monte Carlo classica. Tuttavia, anche a fronte della banalità del caso pratico, i risultati non sono quelli corretti;

- Capitolo 7: nella parte finale si esprimono i punti di forza del Quantum Computing in ambito Risk Analysis ed i molteplici svantaggi che caratterizzano il suo attuale utilizzo.

Capitolo 2

I fondamenti della computazione quantistica

La computazione classica fonda i suoi principi sul concetto di *bit*: esso costituisce l'unità di misura con il quale i calcolatori informatici codificano l'informazione. Matematicamente, il bit è una cifra binaria che assume i valori 0, 1. Essi costituiscono il dominio di realizzazione con il quale si classifica uno stato logico, caratterizzato da due azioni opposte equiprobabili. Naturalmente questo concetto viene generalizzato per sequenze di bit più lunghe dell'unità, in modo tale da poter descrivere sistemi di informazione più complessi.
Negli ultimi decenni del secolo scorso è nata l'idea di codificare l'informazione tramite un'unità fisica più complessa, che si basa sui principi della meccanica quantistica. Infatti, a partire già dagli anni '70, le ricerche della comunità scientifica, tra cui spiccano i nomi dei fisici statunitensi Paul Benioff e Peter Shor, condussero allo sviluppo della teoria quantistica dell'informazione. In questo ambito le ricerche furono promosse dalla accreditata ipotesi che i computer quantistici avrebbero potuto dominare i corrispettivi classici. Nella fattispecie, la comunità scientifica aveva teorizzato che gli elaboratori quantistici avrebbero potuto effettuare simulazioni, la cui replicazione sarebbe stata impossibile anche per i supporti di calcolo classici più sofisticati.

Innanzitutto, per cominciare l'immersione nel dominio del Quantum Computing, è opportuno conoscere il *qubit*.

Definizione 2.0.1. *Il qubit è la più piccola parte di informazione che può essere descritta tramite un sistema quantistico. Infatti, esso indica il quanto di informazione, ossia l'unità di misura informatica, che viene governata tramite le leggi della meccanica quantistica.*

Non è lo scopo di questa tesi fornire le dimostrazioni ai concetti e ai teoremi che risiedono alla base della teoria della fisica quantistica. Però è fondamentale darne un'illustrazione precisa, seppur concisa, in quanto, al fine di comprendere il funzionamento del Quantum Computing e del suo utilizzo, è necessario conoscere i principi che lo governano.
Prima di affrontarli, è opportuno fornire una definizione generale di Quantum Computer.

Definizione 2.0.2. *Un computer quantistico è un calcolatore che esegue operazioni basandosi sul comportamento di particelle subatomiche. Esso sfrutta fenomeni quanto-meccanici come entanglement e superposition.*

Nelle prossime sezioni vengono chiariti i concetti che sono stati menzionati nella precedente definizione. Ma prima di ciò, sono riportate alcune definizioni che risultano fondamentali per il prosieguo dell'analisi.

Definizione 2.0.3. *Uno spazio vettoriale $\mathcal{H}$ si dice spazio di Hilbert sui complessi se è dotato di un prodotto interno*

$$\langle\psi|\theta\rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}, \tag{2.1}$$

definito per ogni coppia di vettori dello spazio ψ e θ. Il prodotto scalare definito sullo spazio $\mathcal{H}$ rispetta le proprietà di positività, linearità ed antisimmetria cioè

$$\langle\psi|\theta\rangle = \langle\psi|\theta\rangle^{\dagger} \tag{2.2}$$

Inoltre, esso induce una norma tale che dato un vettore $\psi \in \mathcal{H}$ è definita come

$$||\psi|| = \langle\psi|\psi\rangle^{1/2} \tag{2.3}$$

tale da rendere $\mathcal{H}$ completo.

Definizione 2.0.4. *Uno spazio vettoriale è completo quando ogni successione di Cauchy di elementi dello spazio converge ad un elemento appartenente allo spazio stesso.*

2.1 I postulati

I *postulati della meccanica quantistica* costituiscono i pilastri teorici su cui si basa la teoria dei sistemi fisici quantistici. Esso possono essere espressi tramite le seguenti asserzioni:

1. **Gli stati**: ogni sistema fisico è descritto biunivocamente da uno stato quantistico rappresentato da un vettore $|\psi\rangle$ di norma unitaria. Esso giace all'interno di un associato spazio di Hilbert $\mathcal{H}$ separabile su campo complesso;

2. **La dinamica**: l'evoluzione del sistema in due istanti di tempo successivi t_1 e t_2 è definita tramite un operatore unitario U, nel seguente modo:

$$|\psi'\rangle = U(t_1, t_2)\,|\psi\rangle\,, \tag{2.4}$$

   dove $|\psi'\rangle\,, |\psi\rangle \in \mathcal{H}$, ossia

$$U : \mathcal{H} \to \mathcal{H}. \tag{2.5}$$

   Questo operatore è definito da una matrice hermitiana, cioè una matrice autoaggiunta. Quest'ultima si ottiene eseguendo la trasposizione di U ed infine sostituendo ogni sua entrata con il valore coniugato. Inoltre deve valere

$$U^\dagger U = I, \tag{2.6}$$

   permettendo ad un vettore di stato di essere trasformato linearmente in un altro vettore ancora unitario rispetto alla norma indotta dal prodotto scalare;

3. **Le osservabili ed il measurement**: una grandezza del sistema fisico viene definita *osservabile*. Per conoscere il suo valore è necessario attuare un'operazione definita *measurement*, che è irreverisibile. Questa è descritta da una successione di operatori M_m di proiezione sullo spazio degli stati. In ambito classico, essa non ha un equivalente, perchè uno stato del sistema è

sempre osservabile con certezza senza che vi sia la necessità di operare una misurazione. Basti pensare che è immediato verificare se un bit è in uno stato 0 oppure 1. Invece, in ambito quantistico il measurement è un procedimento che ha senso in termini probabilistici: il risultato che ne deriva è

$$\frac{M_m |\psi\rangle}{\sqrt{p(m)}} \tag{2.7}$$

dove $p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle$ è la probabilità che lo stato del sistema assuma quello specifico risultato. In altre parole, un'osservabile è definita da un operatore autoaggiunto e l'insieme dei valori che essa può assumere dopo una misurazione è inclusa nel suo spettro;

4. **I sistemi composti**: la descrizione dello stato di un sistema quantistico è descritto dalla combinazione degli n singoli stati dei sottosistemi che lo compongono:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle \tag{2.8}$$

dove $\otimes$ è la notazione del prodotto tensoriale e $|\psi_i\rangle$ è lo stato dell'i-esimo sottosistema.
Allo stesso modo vale per gli spazi vettoriali e cioè: dati due spazi di Hilbert $\mathcal{H}_1$ e $\mathcal{H}_2$ che si riferiscono a due sottosistemi isolati, l'unione di tali sottoinsiemi possiede come spazio degli stati lo spazio di Hilbert ottenuto dalla seguente operazione:

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2. \tag{2.9}$$

I postulati costituiscono le leggi fondamentali della teoria quantistica e risultano utili per ricavare una traduzione matematica dai fenomeni che caratterizzano un sistema fisico-meccanico. A questo punto, è importante riflettere su un concetto che il secondo ed il terzo assioma mettono in luce: mentre la trasformazione di un sistema risulta essere un'operazione deterministica, invece il measurement è probabilistico. Infatti esso non permette di conoscere con certezza quale sarà l'output finale, bensì ne fornisce solamente una distribuzione probabilistica. Ed è questo l'aspetto che maggiormente discosta la teoria quantistica da quella classica. La trasformazione di uno stato in un altro consente di trattare un sistema e di

mantenere tutte le possibili informazioni che esso conserva al suo interno in un unico oggetto che è il vettore di stato. Qualora lo sperimentatore volesse conoscere tali informazioni, egli deve effettuare una misurazione con la quale viene restituito un output con una certa probabilità. Dopodiché lo stato del sistema viene collassato su di esso in modo irreversibile.

2.2 Il qubit

La definizione generale di qubit è stata posta come introduzione a questo capitolo. Il qubit è la più piccola unità fisica che possiede la capacità di codificare informazione. Esso è rappresentato da un vettore all'interno di uno spazio di Hilbert $\mathcal{H}$.

Definizione 2.2.1. *La rappresentazione geometrica del qubit è definita da un vettore, chiamato vettore di stato, la cui formulazione più generica è la seguente:*

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \alpha, \beta \in \mathbb{C}, \tag{2.10}$$

tale che $|\alpha|^2 + |\beta|^2 = 1$.

Dunque, il qubit è una combinazione lineare complessa di due stati limite dello spazio degli stati, identificati dai seguenti vettori:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Questi indicano i due stati logici che formano il dominio di appartenenza del bit classico e, insieme, formano la base ortonormale principale dello spazio del qubit, chiamata *base computazionale*. Dunque, visto nella sua forma più generale, come descritto nella precedente formula, è possibile che lo stato del sistema di un singolo qubit assuma una configurazione intermedia tra questi due valori, raggiungendo uno *stato di sovrapposizione*.

Definizione 2.2.2. *Il fenomeno di superposition è tipico della teoria meccanico-quantistica. Se un sistema dinamico si trova in superposition, significa che esso ha la possibilità di trovarsi in uno o più stati del suo dominio contemporaneamente [3].*

Usualmente, viene fatto ricorso al fenomeno di sovrapposizione per conservare in un solo oggetto fisico tutta l'informazione che è contenuta all'interno di un sistema fisico. Ogni possibile stato che può assumere il qubit a seguito di una misurazione rappresenta una parte di informazione, che è conservata contemporaneamente nello stesso oggetto.
Uno stato di superposition non è osservabile direttamente, nel senso che, dato un qubit nella forma (2.10), non è possibile risalire ai valori esatti di α e β. Il medesimo concetto non ha un corrispettivo in ambito classico, dove, invece, lo stato di un bit è sempre osservabile in uno dei suoi due valori 0 oppure 1 con esattezza. In meccanica quantistica, invece, in linea con quanto i postulati descrivono, per conoscere se lo stato del qubit è $|0\rangle$ oppure $|1\rangle$ bisogna effettuare una misurazione che comporta il primo risultato con probabilità $|\alpha|^2$ oppure il secondo con probabilità $|\beta|^2$, rispettivamente.
Il cosiddetto processo di misurazione mappa lo stato di superposition del qubit in uno dei due vettori appartenenti alla base computazionale. Una volta effettuato, lo stato di sovrapposizione del qubit viene definitivamente abbandonato e non è più possibile ricrearlo. Si potrebbe dire che l'informazione contenuta all'interno di un qubit è uguale a quella di un bit classico, ma si può ottenere solamente dopo una misurazione che ne cambia irreversibilmente lo stato. Pertanto un measurement va effettuato solamente se è necessario fornire un output del sistema. Tuttavia questa procedura non fornisce un output con probabilità quasi certa, in accordo con quanto asseriscono i postulati.
Quindi, supponendo che lo stato inziale sia $|\psi\rangle$, la probabilità che il measurement produca lo stato $|x\rangle$ è data da:

$$p(|x\rangle) = |\langle\psi|x\rangle|^2 \tag{2.11}$$

in cui la notazione $\langle\psi|$ corrisponde al trasposto coniugato del vettore $|\psi\rangle$.

Definizione 2.2.3. *I coefficienti complessi che costituiscono le entrate del vettore di stato sono chiamati amplitudes. Essi moltiplicano ciascuno stato della base*

computazionale. Presi singolarmente, I moduli delle amplitudes elevati al quadrato costituiscono la distribuzione di probabilità discreta dell'output di un processo di measurement.

Dato un vettore di stato, la somma totale dei moduli al quadrato di ciascuna entrata deve rispettare l'assioma probabilistico dell'evento certo. Da ciò consegue la condizione di normalizzazione:

$$\sqrt{|\alpha|^2 + |\beta|^2} = 1, \tag{2.12}$$

dove α e β sono le entrate del vettore di stato del sistema meccanico formato da un qubit.

A questo punto è utile fornire un esempio: dato un sistema fisico ad un qubit, il cui vettore di stato è

$$|q\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} |0\rangle + \frac{i}{\sqrt{2}} |1\rangle$$

risulta facilmente verificabile che $\left|\frac{1}{\sqrt{2}}\right|^2 + \left|\frac{i}{\sqrt{2}}\right|^2 = 1$. Se si effettuasse una misurazione sullo stato di sovrapposizione del qubit, si otterrebbe lo stato $|1\rangle$ con probabilità $\frac{1}{2}$. Tale probabilità è misurabile in questo modo:

$$p(|1\rangle) = |\langle q|1\rangle|^2 = \left|\frac{1}{\sqrt{2}} \langle 0|1\rangle - \frac{i}{\sqrt{2}} \langle 1|1\rangle\right|^2 = \left|-\frac{i}{\sqrt{2}}\right|^2 = \frac{1}{2} \tag{2.13}$$

2.3 La sfera di Bloch

La rappresentazione analitica di un singolo qubit è resa più chiara grazie ad una visualizzazione geometrica. In seguito verrà affrontata la trattazione di un sistema più complesso, derivato dal prodotto tensoriale di più unità quantistiche: per questo tipo di sistema non è possibile ricavare una rappresentazione geometrica del suo insieme. Invece, è sempre possibile definire la rappresentazione geometrica di un singolo qubit.

Definizione 2.3.1. *La sfera di Bloch è una superficie a sfera immersa in uno spazio tridimensionale con raggio unitario. Gli infiniti punti che ne costituiscono*

il luogo geometrico sono posti in relazione biunivoca con le combinazioni lineari complesse che determinano lo stato di superposition di un qubit.

Per descrivere la costruzione dello spazio in cui è centrata la sfera, è necessario partire dalla forma analitica di un qubit. Infatti, anche se dalla misurazione non è possibile determinare le amplitudes, però si può risalire alla loro differenza in fase. Perciò, anziché $\alpha, \beta \in \mathbb{C}$, si costringano al campo dei numeri reali $\mathbb{R}$, e si dica che:

$$|\psi\rangle = \alpha\,|0\rangle + e^{i\phi}\beta\,|1\rangle \tag{2.14}$$

dove $\phi, \alpha, \beta \in \mathbb{R}$. A questo punto, al fine di imporre la condizione di normalizzazione, si può sfruttare la relazione trigonometrica

$$\sin^2 x + \cos^2 x = 1, \tag{2.15}$$

ottenendo così una rappresentazione delle amplitudes in funzione di un angolo θ:

$$\alpha = \cos\left(\frac{\theta}{2}\right) \qquad \beta = \sin\left(\frac{\theta}{2}\right). \tag{2.16}$$

Così facendo, la rappresentazione analitica di un qubit viene espressa in funzione di $\phi, \theta \in \mathbb{R}$:

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,, \tag{2.17}$$

dove $0 \leq \theta \leq \frac{\pi}{2}$ e $0 \leq \phi \leq 2\pi$.
In questo modo, risulta evidente come la sfera possa essere rappresentata nello spazio tridimensionale euclideo $\mathbb{R}^3$. Un generico punto della superficie che identifica biunivocamente lo stato del sistema ha coordinate:

$$\begin{cases} x = \sin 2\theta \cos\phi \\ y = \sin 2\theta \sin\phi \\ z = \cos 2\theta \end{cases}$$

Questa realizzazione è biunivoca per ciascun punto, cioè data una coppia di (θ, ϕ) è possibile risalire ad un punto sulla sfera e viceversa, ad eccezione però dei due punti cartesiani (0,0,1) e $(0,0,-1)$. Questi costituiscono la rappresentazione dei due vettori ortogonali $|0\rangle$ e $|1\rangle$, rispettivamente. Si faccia riferimento alla

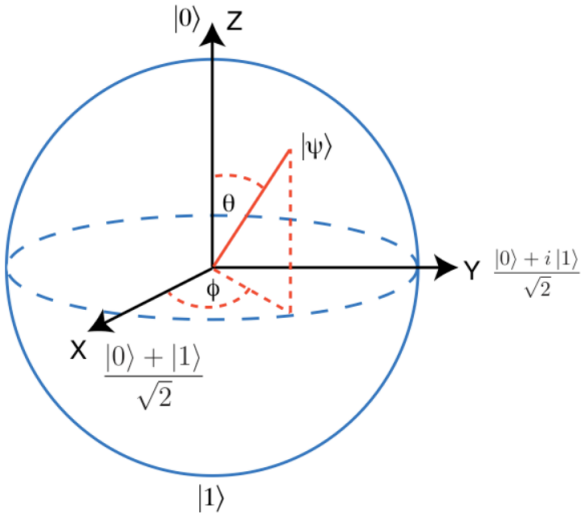


Figura 2.1: Rappresentazione geometrica dello stato $|\psi\rangle$ all'interno della sfera di Bloch.

Figura 2.1: essi individuano sulla sfera i due poli opposti sull'asse verticale Z. In generale, due punti diametralmente opposti costituiscono due vettori ortogonali. Un generico vettore di stato $|\psi\rangle$, associato al sistema fisico, è immerso nella sfera di Bloch. Esso è caratterizzato da un vettore unitario ed è individuato dai due angoli θ e ψ. Nella stessa figura si nota chiaramente che il polo superiore e quello inferiore sono individuati dai due vettori della base computazionale, infatti $|z\rangle = |0\rangle = -|z\rangle = -|1\rangle$. Inoltre, gode di una certa rilevanza anche il vettore $\hat{x}$ che è indicato con la seguente notazione:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Questo, insieme al vettore diametralmente opposto, forma una base ortonormale detta *base secondaria*.

E' importante osservare che il concetto di *statevector*, o stato del qubit, non deve essere confuso con la sua rappresentazione geometrica nella sfera di Bloch. Infatti, nel primo caso si tratta di un vettore bidimensionale giacente nel campo dei numeri complessi. Invece, nel secondo caso, esso è un vettore immerso in uno spazio tridimensionale, definito da componenti reali e funge solo come una rappresentazione geometrica di un singolo qubit.

2.4 Le porte quantistiche

Il secondo postulato della meccanica quantistica descrive la dinamica di un sistema meccanico. Infatti, la computazione quantistica esegue una successione di trasformazioni lineari del vettore di stato, modificando contemporaneamente la probabilità di output con cui l'informazione è codificata su ciascun componente dello stato di superposition.

Definizione 2.4.1. *La trasformazione lineare $U(t_1, t_2)$, che un sistema fisico subisce tra due istanti di tempo distinti t_1 e t_2, è un isomorfismo dello spazio di Hilbert $\mathcal{H}$ in cui è definito il sistema:*

$$U(t_1, t_2) : \mathcal{H} \to \mathcal{H}. \tag{2.18}$$

La matrice U con cui è rappresentata l'operazione è hermitiana, o autoaggiunta, e unitaria, ossia, tale che:

$$U = U^\dagger \qquad U^\dagger U = UU^\dagger = I. \tag{2.19}$$

La teoria dell'algebra lineare stabilisce che una matrice autoaggiunta è normale, ossia:

$$U^\dagger U = UU^\dagger. \tag{2.20}$$

Tali proprietà garantiscono che il vettore di stato abbia norma unitaria, cioè risulti appartenente alla superficie della sfera di Bloch. Una traduzione in senso probabilistico di ciò, comporta che la somma delle probabilità di tutti i possibili stati che esso può assumere sia uguale ad 1.
Dunque, definito il vettore di stato $|\psi\rangle$ al tempo t_1 e dato l'operatore $U(t_1, t_2)$, il nuovo vettore risultante $|\psi\rangle'$ si ottiene con il seguente prodotto vettoriale:

$$|\psi\rangle' = U\,|\psi\rangle\,. \tag{2.21}$$

Gli operatori che agiscono in tal modo vengono definiti *gates*, o anche *porte*.
Di seguito vengono riportati i principali esempi di porte quantistiche che alterano lo stato di un sistema fisico ad un singolo qubit:

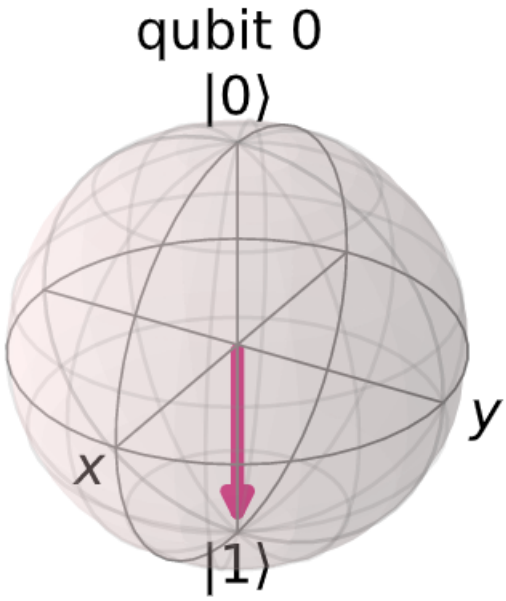
- *X - Gate*: (o anche *Pauli-X Gate*) è definita dalla seguente matrice:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0| .$$

Ad esempio, se lo stato inziale è $|0\rangle$, allora:

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

Di seguito viene mostrato il relativo circuito quantistico ed il risultante vettore di stato nella sfera di Bloch:



- *Y - Gate*:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0| .$$

- *Z - Gate*:
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|.$$

- *Hadamard*: è uno dei più importanti operatori quantistici, in quanto permette di trasportare lo stato inziale di un qubit, di solito inizializzato a $|0\rangle$ oppure $|1\rangle$, in superposition.
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$
$$H|0\rangle = |+\rangle \qquad H|1\rangle = |-\rangle$$

- R_ϕ: è un operatore parametrico che comporta una rotazione di un angolo $\phi \in \mathbb{R}$ attorno all'asse Z della base computazionale.
$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

In generale, esiste una formulazione più generica per la rappresentazione di una porta quantistica: essa sfrutta l'utilizzo dei parametri reali θ, ϕ e λ (i primi due sono già stati introdotti precedentemente). Infatti, si può dire che la forma matriciale di ciascun operatore può essere ricondotta per determinati valori di tali parametri alla seguente:

$$U_3\left(\frac{\pi}{2}, \phi, \lambda\right) = \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\theta}\sin\frac{\theta}{2} & e^{i\lambda+i\theta}\cos\frac{\theta}{2} \end{pmatrix}$$

E' importante osservare che ciascun operatore definito in questa sezione costituisce la formulazione matematica delle osservabili. Il terzo postulato asserisce che una misurazione di una osservabile comporta un risultato, secondo una determinata probabilità, che è contenuto all'interno dello spettro dell'operatore che la caratterizza. Dato uno stato $|i\rangle$ e l'operatore hermitiano A, l'equazione da cui si ricava il corrispondente autovalore λ_i è

$$A|i\rangle = \lambda_i |i\rangle. \tag{2.22}$$

Ogni operatore normale autoaggiunto A può essere rappresentato tramite l'espansione del suo spettro tramite la seguente equazione:

$$A = \sum_i \lambda_i |i\rangle \langle i| , \tag{2.23}$$

dove $|i\rangle \langle i|$ è l'operatore di proiezione sull'i-esimo autostato di A. Il carattere hermitiano della matrice garantisce che gli autovalori siano reali. Questa formulazione risulta utile quando è necessario valutare funzioni dell'osservabile. In tal modo:

$$f(A) = \sum_i f(\lambda_i) |i\rangle \langle i| . \tag{2.24}$$

Si consideri, ad esempio, l'operatore X. I relativi autovettori sono ricavati attraverso semplici passaggi di algebra lineare:

$$det\,(X - \lambda I) = \lambda^2 - 1 = 0 \Rightarrow \lambda = \pm 1. \tag{2.25}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \Rightarrow a = b$$

$$|a|^2 + |b|^2 = 1 \Rightarrow a = b = \frac{1}{\sqrt{2}}$$

Allo stesso modo si ricava il secondo autovettore. Dunque, i due autostati dell'operatore X sono:

$$|+\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \quad |-\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} .$$

Quindi, utilizzando gli operatori di proiezione

$$P_1 = |+\rangle \langle +| \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \quad P_2 = |-\rangle \langle -| = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} ,$$

si ottiene la matrice originale:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = P_1 - P_2.$$

Gli autostati che costituiscono il dominio di misurazione dell'osservabile non fanno parte della base computazionale $|0\rangle , |1\rangle$, bensì quella secondaria. In generale, il

processo di misurazione può essere eseguito rispetto ad infinite basi dello spazio di Hilbert. La scelta di tale insieme di vettori ortogonali è arbitraria. Tuttavia, quella che produce il risultato di riferimento per la computazione quantistica è quella computazionale. Essa costituisce il dominio di codifica.
La distribuzione di probabilità tale per cui si misura uno stato $|\psi\rangle$ nella base secondaria, detta anche *X-basis*, come quella del precedente esempio, è la seguente:

$$p(|+\rangle) = |\langle +|\psi\rangle|^2 \quad p(|-\rangle) = |\langle -|\psi\rangle|^2. \tag{2.26}$$

Dopo aver effettuato il measurement, lo stato del qubit coincide con uno dei due autostati della matrice X. Se è necessario fornire un output rispetto alla base computazionale, basta eseguire un measurement rispetto a tale base. Se si misura uno dei due autostati della matrice X, si ottiene uno dei due vettori $|0\rangle, |1\rangle$ con probabilità uguale a $\frac{1}{2}$, cioè del tutto casuale. Ciò evidenzia la differenza di misurazione nelle due differenti basi $Z-basis$ e $X-basis$, tale per cui esistono due precisi vettori la cui misurazione rispetto alla prima base forniscono uno dei due autostati con probabilità 1, mentre misurati rispetto all'altra ha casualità massima.
Una caratteristica tipica delle porte quantistiche è l'*universalità*.

Definizione 2.4.2. *La proprietà di universalità garantisce che tutti gli stati dello spazio di Hilbert associato ad un generico sistema fisico sono raggiungibili tramite la sequenza di una o più porte quantistiche.*

2.5 Gli stati a qubits multipli

Fino ad ora è stato trattato il caso di un singolo qubit, che costituisce il sistema fisico meccanico più semplice possibile. Esso è rappresentato da due amplitudes e può rappresentare due singoli stati dopo il measurement.
In realtà, lo stato di un sistema fisico è generalmente rappresentato da un vettore di stato dimensionalmente più grande rispetto a quello trattato nei capitoli precedenti. Questo ovviamente dipende dalla complessità del problema che è interesse di studio. In termini pratici, questo si traduce in un vettore di stato che è rappresentato da una molteplicità di qubit. Di conseguenza esso giace in uno spazio di Hilbert di

dimensione maggiore.
Ad esempio, con un sistema di due qubits, il numero di amplitudes totale è quattro (due per ciascun qubit) e lo è anche il numero degli stati che sono i seguenti:

$$|00\rangle \quad |01\rangle \quad |10\rangle \quad |11\rangle . \tag{2.27}$$

Il vettore di stato ψ appartiene a uno spazio $4D$:

$$\psi = a_{00}\,|00\rangle + a_{01}\,|01\rangle + a_{10}\,|10\rangle + a_{11}\,|11\rangle \tag{2.28}$$

dove i coefficienti $a_{00}, a_{01}, a_{10}, a_{11}$ di ciascun vettore si riferiscono alle rispettive amplitudes. Anche in questo caso, vale la regola di normalizzazione tale per cui

$$|a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1. \tag{2.29}$$

Ciò significa che, preso il vettore di stato del sistema, la probabilità che un measurement restituisca come output lo stato $|01\rangle$ è uguale a $|a_{01}|^2$. Dunque, continuano a valere le regole del measurement allo stesso modo:

$$p(|01\rangle) = |\langle\psi|01\rangle|^2. \tag{2.30}$$

La regola di rappresentazione del vettore di stato del sistema multiqubits, dati i singoli qubits

$$|a\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \quad |b\rangle = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}$$

interagenti è

$$|\psi\rangle = |b\rangle \otimes |a\rangle = \begin{pmatrix} b_0 \times \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \\ b_1 \times \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} b_0 \times a_0 \\ b_0 \times a_1 \\ b_1 \times a_0 \\ b_1 \times a_1 \end{pmatrix}.$$

Risulta immediato ricavare il risultato per il caso di tre qubits a, b, c, in cui lo statevector è dato da:

$$\begin{pmatrix} c_0 \times b_0 \times a_0 \\ c_0 \times b_0 \times a_1 \\ c_0 \times b_1 \times a_0 \\ c_0 \times b_1 \times a_1 \\ c_1 \times b_0 \times a_0 \\ c_1 \times b_0 \times a_1 \\ c_1 \times b_1 \times a_0 \\ c_1 \times b_1 \times a_1 \end{pmatrix}.$$

Generalizzando il problema a n qubits disponibili, è necessario tenere traccia di 2^n amplitudes totali, perchè esso è anche il numero di stati che può assumere il sistema meccanico in questione. Questo comporta la sostanziale differenza con gli elaboratori classici, per i quali risulta assai ardua la simulazione del comportamento di un computer quantistico. La causa è fatta coincidere con l'elevato numero di stati che esso può assumere. Ad esempio, con 20 qubits, questo numero supera già il milione.

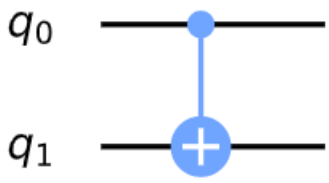
Dopo la definizione dei multi-qubits, occorre affrontare i gates che operano con più di un qubit alla volta. Queste porte quantistiche mettono in relazione più qubits tra di loro, ma mantengono le stesse caratteristiche già menzionate precedentemente e che ogni gates deve soddisfare. Ovviamente, la loro dimensione dipende da quella del vettore di stato. La principale tra queste prende il nome di *CNOT - Gate*. Il nome per esteso è *conditional not gate* e agisce su due qubits. Al primo di essi, definito *target*, viene applicato un X solo se il secondo, definito *control*, ha stato $|1\rangle$. La matrice che ne regola il funzionamento è

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

se il qubit di controllo è il primo dei due, oppure

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

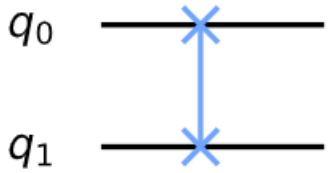
se corrisponde al secondo. La rappresentazione grafica di tale porta in un circuito implementato su Qiskit è mostrata nella seguente figura: dove il secondo qubit è il



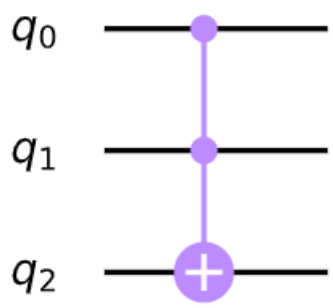
target.
Un altro multi gate notevole è il cosiddetto *Swap*:

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

Esso scambia le amplitudes tra i due qubits e viene indicato dal simbolo:



Per finire con gli esempi notevoli di porte multiqubits, è utile definire il *Toffoli Gate*. Esso opera su tre qubits ed è per questo che viene anche detto anche *CCNOT*. Infatti, Toffoli agisce come è stato definito per CNOT, ma con un doppio controllo,



anzichè singolo. Come si vede dal simbolo raffigurato sul circuito, esso compie un X gate sul terzo qubit, che è il target, se i precedenti qubits di controllo sono entrambi in stato $|1\rangle$.

2.6 Bell State ed entaglement

Definizione 2.6.1. *Lo Stato di Bell è uno stato notevole di un sistema quadridimensionale, cioè composto da due qubits. La sua forma analitica viene espressa con la seguente formulazione:*

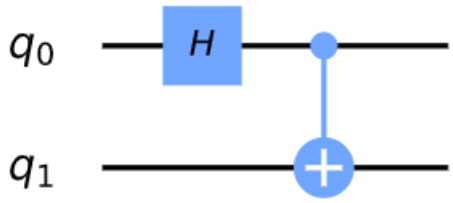
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.31}$$

Il vettore di stato di Bell è

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

ed è facilmente ottenibile tramite l'azione combinata degli operatori Hadamard al primo qubit, e di CNOT con target il secondo qubit:

$$CNOT((H\,|0\rangle) \otimes |0\rangle) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \left(\left(\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$



Come si può osservare, esso è una combinazione lineare dei due stati $|00\rangle$ e $|11\rangle$ opposti, e le loro amplitudes determinano che essi sono equiprobabili a seguito di un processo di misurazione. Si dice che i due qubits sono messi in massima correlazione. Infatti, il Bell State viene sfruttato per introdurre un fenomeno fondamentale della meccanica quantistica che prende il nome di *entanglement*.

Definizione 2.6.2. *L'entanglement di un sistema complesso a due o più qubits si verifica quando i loro sottostati sono messi in superposition di stati correlati. Ossia, una volta conosciuti i sottostati di tutti i qubits a meno di uno, allora quest'ultimo risulta direttamente ricavabile [4].*

Infatti, riducendosi al caso più semplice di un sistema fisico composto da due particelle, una volta effettuata una misurazione su un sottostima, è una immediata conseguenza sapere anche il valore dell'altro sottosistema senza effettuarne una misurazione. Einstein teorizzò il concetto di *spooky action at a distance*, in quanto tale fenomeno si mantiene a prescindere dalla distanza fisica che li separa. L'Entanglement condusse inizialmente ad una crisi della teoria quantistica dopo la sua formulazione. Infatti il concetto di trasmissione istantanea di informazione portava erroneamente a contraddire il *principio di località* tale per cui due sistemi abbastanza lontani non possono avere effetti reciproci simultanei. I primi a mettere in luce questo problema furono gli scienziati Albert Einstein, Boris Podolsky e Nathan Rosen. Successivamente, insieme dimostrarono, tramite un esperimento mentale, come in realtà la teoria della meccanica quantistica non fosse errata, bensì incompleta [5]. Esistono pertanto delle variabili nascoste che governano il fenomeno dell'entanglement e che solo studi futuri avrebbero potuto svelarne il

corretto funzionamento.

2.7 I circuiti quantistici

Una volta descritta la teoria del QC e gli attori che ne entrano in gioco, è opportuno definire qual è il collegamento che li lega con la computazione quantistica.

Definizione 2.7.1. *Un circuito quantistico è un processo computazionale che riceve in input uno stato iniziale e fornisce in output il risultato di una misurazione. Esso è composto da un insieme di corde, che fanno riferimento ai qubits del sistema, e da una serie consecutiva di gates, attraverso cui è possibile compiere opportune trasformazioni lineari del vettore di stato in superposition.*

Le parti di cui è composto un circuito seguono un preciso ordine sequenziale, di cui si fornisce l'elenco nel seguito:

1. **inizializzazione**: in un primo momento si definisce il numero di qubits che viene utilizzato per la simulazione dell'esperimento di interesse. Ogni qubits è rappresentato da una linea orizzontale e la convenzione esige che esso sia inizializzato a zero. Se risultano necessari, può essere fatto uso di ulteriori bits classici o qubits chiamati *ancillas*, che servono per la memorizzazione intermedia di alcune grandezze fisiche, o per l'esecuzione di particolari algoritmi classici. Inoltre, siccome alla conclusione di ogni circuito c'è una fase di measurement, è opportuna la definizione dei bits classici su cui viene conservato l'output binario finale di ogni misurazione. Esso corrisponde ad uno tra i valori 0 e 1. Di solito i dati iniziali, che essi siano conservati su qubits o bits, vengono raggruppati in *registri*;

2. **trasformazione**: segue una fase di manipolazione dello stato, in cui intervengono i gates quantistici che compiono opportune trasformazioni sul sistema generale. Tutto ciò avviene in base al fine richiesto dal circuito quantistico. Di solito intervengono successioni di gates atti ad approssimare note funzioni chiamate *oracles*, di cui si darà una definizione in questo capitolo. Questa

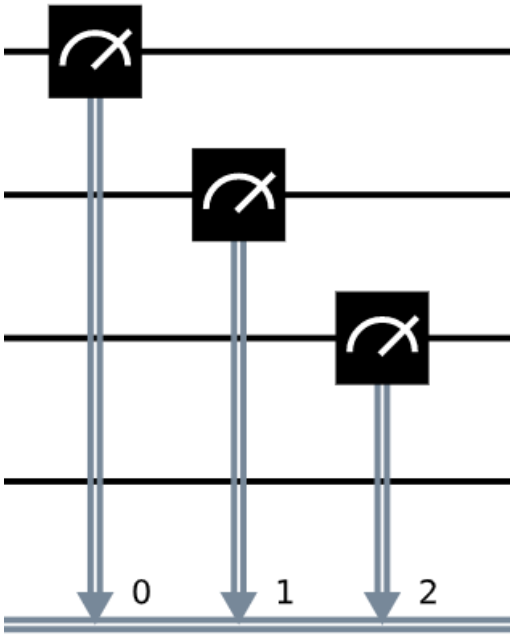


Figura 2.2: In questo esempio i primi tre qubit del registro subiscono un measurement su altrettanti bit classici, i quali sono riuniti in un unico registro rappresentato da un'unica doppia linea continua.

pratica consente di ricreare lo stato quantistico richiesto dal problema, nella fase precedente alle misurazioni;

3. **misurazione**: la fase finale è quella in cui viene eseguito il processo del measurement, il quale possiede natura probabilistica. Esso collassa lo stato del sistema su uno dei possibili stati della base computazionale, fornendo così l'output finale. Tale risultato viene conservato sulle corde del registro riservato ai bit classici, che di norma si trovano nella parte inferiore del circuito. Il simbolo della misurazione è riportato in Figura 2.2.

A titolo di esempio, in Figura 2.3 viene riportato un'implementazione di un circuito quantistico. Al suo interno sono riconoscibili le fasi che lo compongono. Per una più facile visualizzazione del circuito, il composer è stato dotato dell'inserimento di *barriere* per permettere la separazione di sezioni differenti dell'algoritmo.

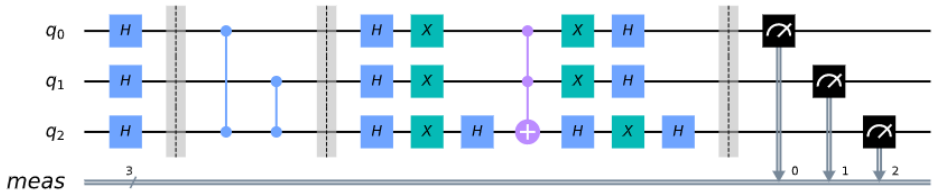


Figura 2.3: Grover's Algortihm è un algoritmo quantistico con il quale si può risolvere un problema di ricerca di uno o più elementi all'interno di un insieme di dati. Spesso viene utilizzato per l'interrogazione di database per la selezione di un determinato elemento, il quale passato in input ad una funzione specifica fornisce un output particolare. In questo esempio il circuito è composto da un registro di tre qubits, tre barriere e nella parte finale sono presenti tre misurazioni.

I simboli e le immagini che fanno riferimento ad un circuito quantistico derivano dalla costruzione di tali oggetti in un ambiente che è reso disponibile nelle piattaforme open-source di IBM. Infatti, come già l'introduzione a questo lavoro fa menzione, la tesi fa riferimento ai risultati delle simulazioni che sono state implementate attraverso gli strumenti informatici messi a disposizione da IBM. Essi sono principalmente tre:

- il cosiddetto *Composer*: è una visualizzazione grafica della manipolazione dello stato del sistema tramite l'utilizzo di porte quantistiche;
- *Qiskit*: è un package scritto in linguaggio Pyhton che permette l'implementazione di sistemi quantistici tramite codice di programmazione;
- *OpenQuasm*: è un linguaggio di istruzioni testuali, nel funzionamento simile a Qiskit.

In questa tesi, lo strumento che viene utilizzato per la sperimentazione di circuiti quantistici è la libreria Qiskit. Essa permette di sviluppare programmi di QC e di eseguirli attraverso specifici simulatori che riproducono le funzionalità di un elaboratore quantistico. Ciò che rende le piattafrome di IBM eccezionali è la possibilità di eseguire un run del proprio codice direttamente sull'hardware quantistico. Esso è reso accessibile da IBM tramite un cloud. Il computer quantistico di IBM si chiama *IBM Q System One* ed è stato presentato negli anni recenti dalla azienda statunitense. Esso possiede un processore di 5 qubits, un numero che permette

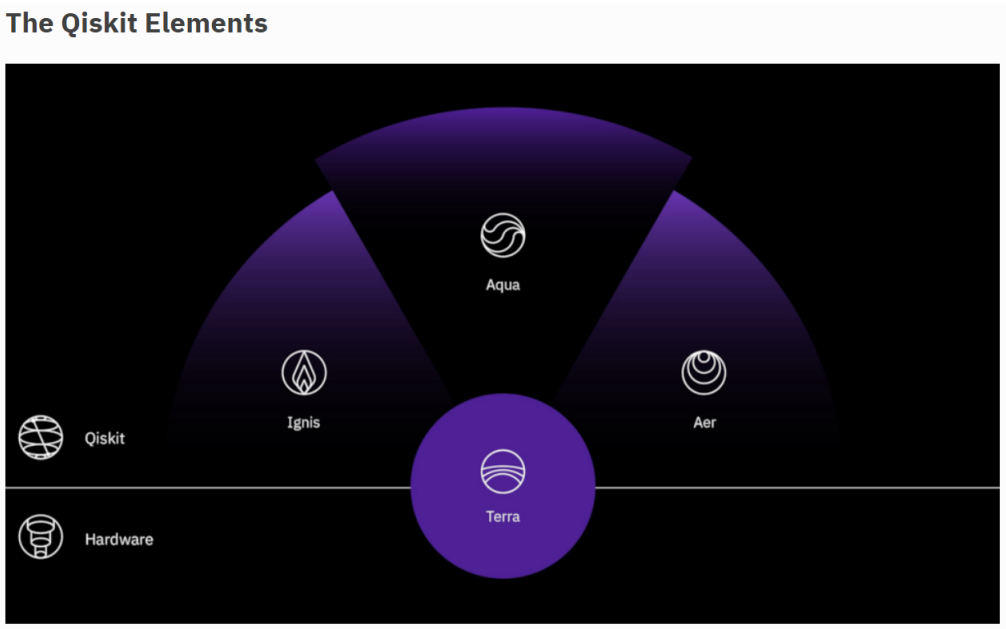


Figura 2.4: La documentazione messa a disposizione su IBM Quantum Experience suddivide Qiskit in quattro moduli: Terra, Ignis, Aer, Aqua.

la rappresentazione di 2^5 stati. Recentemente è stato introdotto una versione a 16 qubits. Il nome di queste macchine deriva dalla sede in cui esse sono locate. L'esecuzione di un codice inviato al cloud non è istantanea, bensì deve rispettare una coda di tutte le utenze della piattaforma.

Qiskit è costituito da quattro moduli di costruzione e manipolazione dei circuiti quantistici:

- **Terra**: è il fondamento su cui si basa tutto il framework e costituisce il linguaggio di basso livello con cui si determinano le primitive che implementano i circuiti e si gestiscono gli impulsi;

- **Ignis**: attraverso questo modulo si gestisce l'errore quantistico delle porte e dei circuiti. Infatti, l'hardware degli attuali computer quantistici non è perfetto, bensì opera con imprecisione. Di conseguenza, essi provocano effetti distorsivi sui risultati ottenuti. Tale errore è chiamato *errore di decoerenza*, di cui si darà una breve illustrazione in conclusione al corrente capitolo;

- **Aer**: è l'insieme dei simulatori e dei debuggers che emulano le operazioni dei computer quantistici. In questo modo si possono ottenere risultati precisi, non affetti da errori di decoerenza. Inoltre, i simulatori riescono a trattare problemi che fanno uso di un numero piuttosto elevato di qubits. Questo invece non è sempre possibile su hardware reali, in quanto i processori attuali sono costituiti da un numero più basso di qubits. In più, i tempi di esecuzione risultano estremamente più veloci. I simulatori disponibili in questo modulo sono lo *Statevector simulator* che permette di simulare tramite una sola realizzazione il vettore di stato del sistema. Invece, *Qasm simulator* opera multiple esecuzioni del vettore di stato finale, con l'emulazione di errore di decoerenza. *Unitary simulator* restituisce l'operatore matriciale unitario che corrisponde all'interno circuito passato in input. Quest'ultimo non fa utilizzo di misurazione;

- **Aqua**: è il modulo con il quale si possono trattare i problemi quantistici attraverso un linguaggio di alto livello. Esso fa uso direttamente delle funzioni dei precedenti moduli, senza necessariamente tradurre il problema in istruzioni di basso livello. Infatti, Aqua contiene un ampio insieme di algoritmi e circuiti già implementati per poter trattare problemi scientifici di diversa natura e che rientrano nei campi di finanza, machine learning, ottimizzazione e chimica. Sono messe a disposizione anche le funzioni che permettono una diretta manipolazione degli strumenti citati.

Nello svolgimento di questa tesi, il modulo che è stato utilizzato è l'ultimo che è stato presentato. Aqua, non a caso, permette di utilizzare circuiti quantistici e funzioni già implementati, che sono reperibili sulle repositories GitHub di Qiskit. Questa strumento è nato infatti con la volontà di rendere accessibile alla comunità scientifica la tecnologia del QC e permettere agli utenti esterni di procedere nella sua indagine e contribuire nella ricerca. Il fine di questa scelta è strategica, in quanto si spera che possa accelerare lo sviluppo delle tecniche quantistiche per la risoluzione dei problemi oggetti di studio.

2.8 Quantum oracles e Phase kickback

I circuiti costituiscono la forma fisica su hardware di algoritmi quantistici. In generale, al loro interno, esistono delle procedure che svolgono precisi ruoli al fine del conseguimento di determinati risultati, anche primari. Spesso, questi oggetti sono comuni a più algoritmi. Dunque, è risultato utile generalizzarli per poterne fornire una descrizione teorica.

Definizione 2.8.1. *Quantum oracle è una macchina astratta O che viene utilizzata per risolvere problemi di tipo decisionale. Essa è collegata ad una funzione booleana black-box f che permette di rispondere ad un determinato parametro passato in input.*

Definizione 2.8.2. *Una funzione black-box f è un operatore di cui non si conoscono le operazioni e i metodi che essa usa per fornire il risultato. Di essa si conoscono solamente i valori di output in relazione al dominio di ingresso.*

Gli oracles sono delle subroutines che usualmente vengono utilizzati nelle fasi iniziali di un circuito. Essi agiscono in determinati problemi decisionali per poter individuare gli input di interesse compresi tra il dominio allargato degli stati totali di ingresso. L'implementazione su circuito quantistico costituisce un decisivo collo di bottiglia, in quanto esso richiede uno sforzo progettuale non trascurabile. Uno dei metodi che viene utilizzato per rappresentarli è definito in funzione della fase che esso compie sullo stato passato in input.

Definizione 2.8.3. *Dato lo spazio di Hilbert $\mathcal{H}$, si dice che $O_f : \mathcal{H} \to \mathcal{H}$ è un phase oracle [6] se, detta $f : X \to [-1,1]$ ed $\{|x\rangle : x \in X\}$ base ortonormale di $\mathcal{H}$, esso agisce nel seguente modo:*

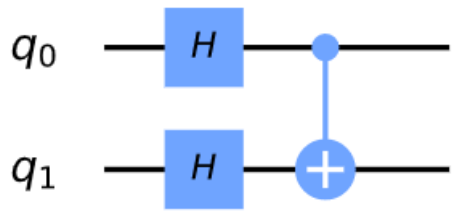
$$O_f : |x\rangle \to e^{if(x)} |x\rangle . \tag{2.32}$$

Questo modello di oracle viene utilizzato per problemi di interrogazione di database, come il Grover's Problem di cui verrà ampiamente analizzato nei capitoli seguenti.

Un secondo elemento chiave che rientra tra le procedure quantistiche, viene spesso utilizzato per l'implementazione degli oracles. Esso è descritto dalla seguente definizione.

Definizione 2.8.4. *Phase kickback è un artificio quantistico che viene applicato su un registro di qubits tramite l'implementazione di uno specifico design circuitale. Esso consente di trasportare una quantità di informazione da un qubit ad un insieme di qubits tramite una serie di operazioni di controllo.*

L'esempio di kickback più semplice riguarda la manipolazione di un circuito a due qubits. Si immagini la seguente situazione:



Viene applicato un operatore CNOT ai due qubits messi in superposition, il primo dei quali funge da controllo:

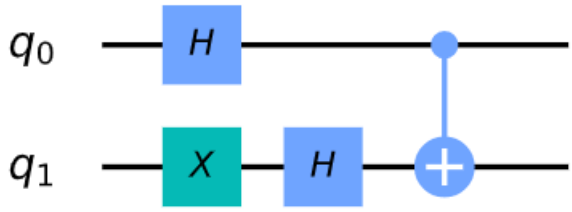
$$CNOT\left(H\left|0\right\rangle \otimes H\left|0\right\rangle\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \left|{+}{+}\right\rangle$$

$$= \left|{+}{+}\right\rangle = \frac{1}{2}\left(\left|00\right\rangle + \left|01\right\rangle + \left|10\right\rangle + \left|11\right\rangle\right).$$

Si applichi alla fine del circuito un'operazione H a ciascun qubits:

$$H\left|+\right\rangle \otimes H\left|+\right\rangle = \left|00\right\rangle.$$

Adesso, si consideri la seguente circostanza:



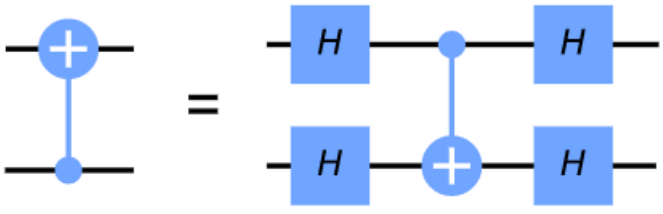
$$CNOT\left(H\left|0\right\rangle \otimes XH\left|0\right\rangle\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \left|-+\right\rangle$$

$$= \left|--\right\rangle = \frac{1}{2}\left(\left|00\right\rangle - \left|01\right\rangle - \left|10\right\rangle + \left|11\right\rangle\right).$$

Come si evince, l'effetto che si ottiene da CNOT in realtà corrisponde all'opposto di come esso è stato definito: infatti, risulta modificato il sottostato del qubit control, mentre quello target rimane invariato. Come nel caso precedente, si applichi H a ciascun qubits:

$$H\left|-\right\rangle \otimes H\left|-\right\rangle = \left|11\right\rangle.$$

Il risultato ottenuto nei precedenti casi è ottenibile anche applicando semplicemente un CNOT al registro di input, in cui il target corrisponde al primo qubit. In altre parole, vale la seguente uguaglianza: Ciò è direttamente verificabile osservando



le corrispondenti matrici di trasformazione, che risultano entrmabe uguali a

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

Questo è l'esempio più semplice di phase kickback: l'effetto CNOT compreso tra due operazioni H viene trasferito dal secondo al primo qubit. In altri termini, la fase generata dal gate CNOT viene trasferita da uno dei due fili del registro all'altro.

2.9 L'errore di decorenza e la NISQ era

La teoria quantistica ha cominicato a progredire a partire dalla prima metà del ventesimo secolo. I postulati, i teoremi ed i meccanismi che la scienza è riuscita a formulare a riguardo sono sfociati da un intenso movimento scientifico innovativo. Già allora si immaginava di poter eseguire gli algoritmi quantistici, oppure simulare fenomeni tipici di questo campo di sperimentazione, tramite l'utilizzo di macchine ideali *fault-tolerant*. In realtà, i primi elaboratori vennero sviluppati qualche decennio più tardi ed ancora oggi essi non sono esenti da imprecisioni di calcolo.

Definizione 2.9.1. *La decoerenza è un fenomeno tipico dei calcolatori quantistici. Essa indica la circostanza per cui un sistema meccanico isolato subisce l'azione di interferenze provenienti dall'esterno, il cui effetto si traduce in un decadimento dello stato del sistema. Si dice, infatti, che lo stato non è coerente, quando le grandezze misurate sulle funzioni d'onda delle particelle che costituiscono il sistema rompono l'entanglement, cioè la correlazione quantistica che le lega.*

Gli attuali hardwares quantistici sono macchine imperfette, cariche di errore. Al fine di ottenere un risultato il cui errore può essere ritenuto preciso, è opportuno che tutti i calcoli vengano eseguiti all'interno del tempo di decoerenza. Inoltre, all'aumento del numero di qubits e delle porte all'interno del circuito corrisponde una diminuzione di tale orizzonte temporale, nonchè un aumento notevole del

rumore generale.

John Preskill, fisico statunitense, ha espresso in [7] lo stato dell'arte del QC e ne ha delineato le potenzialità. Soprattutto, si è preoccupato di descrivere le sfide future che la scienza dovrà affrontare nei prossimi anni. Egli ha inoltre definito il concetto di *Noisy Intermediate-Scale Quantum* era. Nei decenni a venire diverranno comuni le tecnologie quantistiche che permetteranno l'implementazione di circuiti quantisici, i quali saranno costituiti da un numero modesto di qubits. Infatti, per Intermediate-Scale si intende circa 50-100 qubits, i quali permetteranno la rappresentazione di un numero non troppo basso di stati totali, al contrario di quelli attuali. Tuttavia, essi saranno ancora carichi di rumore e risulteranno in grado di accettare esclusivamente gli algoritmi che richiedono un design circuitale di ristrette profondità. Pertanto, l'obbiettivo che la scienza deve porsi nel futuro prossimo è l'incremento del numero di qubits dei processori quantistici, ma allo stesso tempo garantire un livello di qualità maggiore dell'esecuzione.

Capitolo 3

Gli algoritmi quantistici

La teoria della computazione quantistica comprende un insieme piuttosto ristretto di algoritmi che sono stati sviluppati a partire dalla seconda metà del secolo scorso. Essi sono nati per tentare di proporre metodi di soluzione alternativi a quelli classici per problemi matematici noti. In generale, mentre un algoritmo classico può essere eseguito su un computer quantistico, invece non è sempre vero il contrario. Infatti, ciò che contraddistingue un algoritmo quantistico da uno classico è l'utilizzo dei fenomeni di measurement, entanglement e superposition, che non trovano un corrispettivo significato in un contesto classico.
Un algoritmo quantum viene eseguito su un circuito per la risoluzione di un problema. Pertanto, esso richiede una specifica implementazione dei gates che lo costituiscono. Infatti, per capire il funzionamento di uno specifico algoritmo, è importante capire qual è il procedimento di codifica dei dati reali all'interno del sistema quantistico dell'hardware.
Due degli algoritmi più importanti sono lo *Shor's Algorithm* e il *Grover's Algorithm*. Essi sono chiamati con il nome dei loro scopritori, l'americano Peter Shor e l'indiano-americano Lov Grover. Il primo propose il suo algoritmo nel 1994 [8], tramite il quale egli dimostrò la possibilità di fattorizzare un intero N nei suoi fattori primi in un tempo di esecuzione polinomiale in $\log N$, cioè $\mathcal{O}(\log N)$, anziché esponenziale. La teoria computazionale classica, di contro, sostiene che non esiste un equivalente algoritmo con le medesime potenzialità, in quanto il problema rientra

tra gli *NP-hard*. In altre parole, non esiste un metodo deterministico che produca la soluzione con un numero di istruzioni di ordine polinomiale rispetto alla grandezza del problema. Ovviamente l'algoritmo di Shor ha aperto ad ampie discussioni nel campo della crittografia, in quanto potrebbe essere la chiave per rompere alcuni schemi crittografici, come ad esempio l'RSA, in tempi ragionevolmente ristretti. Tuttavia, sebbene teoricamente questo algoritmo possa avere successo, bisogna tenere presente che lo stato dell'arte del QC lo descrive ancora come in una fase primordiale. Infatti, anche se i simulatori consentono di eseguire sperimentazioni con un numero iniziale di qubits abbastanza elevato, invece, i sistemi quantistici sviluppati finora non contano un numero sufficiente per poter risolvere un caso reale. Nel 2019 IBM ha lanciato il suo IBM Q System One, che consente la simulazione di qualche milione di stati, mentre ora sta lavorando per presentarne una versione che è attrezzata con più di 50 unità. Google, nello stesso anno, ha annunciato che il suo processore a 72 qubits è in fase di sperimentazione. Tuttavia, sebbene presto si arriverà a poter gestire più facilmente problemi a dimensione elevata, la quantità spesso opera a discapito della qualità. Quando l'hardware diventa più complesso, è difficile gestire gli errori quantistici di decoerenza. Pertanto, secondo le grandi multinazionali tecnologiche un importante obbiettivo in questo campo di ricerca prevede di raggiungere una tecnologia che sia in grado di compiere calcoli precisi, ma che allo stesso tempo mitighi l'errore di cui è affetta.

3.1 Creare superpositions per la codifica di distribuzioni probabilistiche

Il funzionamento degli algoritmi del QC verte principalmente sulla distribuzione di probabilità di misurazione del vettore di stato $|\psi\rangle$ associato al sistema. Per questo motivo, essi vengono definiti probabilistici.
Una domanda che è importante porsi a questo punto è la seguente: data una distribuzione di probabilità di un problema reale, come è possibile codificare la sua funzione sul sistema computazionale quantistico?

E' stato dimostrato che per l'insieme delle distribuzioni *log-concave* esiste un metodo efficiente per rispondere a questo quesito [9].

Definizione 3.1.1. *Una funzione* $f : \mathbb{R}^n \to \mathbb{R}_+$ *è logaritmicamente concava se* $\forall x, y \in Dom(f)$ *e* $0 < \theta < 1$*, vale:*

$$f\left(\theta x + (1-\theta)y\right) \leq f(x)^{\theta} f(y)^{1-\theta} \tag{3.1}$$

da cui segue direttamente:

$$\log f\left(\theta x + (1-\theta)y\right) \leq \theta \log f(x) + (1-\theta)\, f(y). \tag{3.2}$$

Ad esempio, rientrano in questa famiglia di funzioni le PDF gaussiane e normali multivariate, esponenziali e uniformi.
Sia p_X una funzione di densità continua associata alla realizzazione di una variabile casuale X. In un sistema quantistico, il suo dominio viene discretizzato in N regioni e ognuna di essa viene associata ad uno degli stati che formano la base ortonormale di misurazione dello spazio di Hilbert $\mathcal{H}$ del sistema. Dunque, vale $N = 2^n$, dove n è il numero di qubits del registro in cui viene caricata la distribuzione. Ovviamente, più questo numero è elevato e minore è l'errore di approssimazione dovuto al processo di discretizzazione.
Per formalizzare questo concetto, si può dire che, per $i = \{0, \ldots, N-1\}$, la p_X è rappresentata da una successione reale $\{p_i\}$, tale che $\sum_i p_i = 1$ e $0 \leq p_i \leq 1$. Il generico valore p_i rappresenta il valore di probabilità associato alla i-esima regione del dominio, a partire da sinistra.
Il processo seguente prevede di creare un appropriato stato di superposition che corrisponde al vettore di stato del sistema meccanico:

$$|\psi\left(\{p_i\}\right)\rangle = \sum_{i=0}^{N-1} \sqrt{p_i}\, |i\rangle\,, \tag{3.3}$$

dove $|i\rangle = |x_1, \ldots, x_n\rangle \in \{0,1\}^n$ è l'i-esimo vettore della base ortonormale di misurazione e x_1 è il valore del suo qubit più significativo.
Per dimostrare che l'integrazione a seguito di questa procedura risulti efficiente, basta seguire un ragionamento induttivo. Si immagini di suddividere il dominio di

p_X in 2^m regioni in modo che valga:

$$|\psi\rangle_m = \sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}\,|i\rangle\,, \tag{3.4}$$

dove $p_i^{(m)}$ è la probabilità associata alla i-esima regione e l'apice (m) tiene conto del numero totale di divisioni. Per ottenere 2^{m+1} regioni, è opportuno ricorrere ad un metodo efficiente per codificare tale informazione su QC. Quindi, si aggiunga un ulteriore qubit al registro iniziale, tale per cui lo stato risulta essere:

$$\sum_{i=0}^{2^m-1} \sqrt{p_i^{(m)}}\,|i\rangle \to \sqrt{\alpha_i}\,|i\rangle\,|0\rangle + \sqrt{\beta_i}\,|i\rangle\,|1\rangle\,, \tag{3.5}$$

in cui α è la probabilità associata alla parte sinistra della regione i e β la probabilità della parte complementare.
Lo stato di nuova creazione è

$$|\psi\rangle_{m+1} = \sum_{i=0}^{2^{m+1}-1} \sqrt{p_i^{(m+1)}}\,|i\rangle\,, \tag{3.6}$$

che corrisponde ancora ad uno stato della forma (3.3).
Una volta dimostrato il modo di implementazione di tale stato, è utile capire come esso possa costituire un punto di partenza per la soluzione di alcuni problemi probabilistici. Uno di questi, per esempio, è il campionamento da una funzione di distribuzione. Tuttavia, per il prosieguo di questo lavoro, lo scopo che si fa dello stato di superposition iniziale non coincide solamente con il campionamento, come si vedrà nel seguito.

3.2 The Grover's Problem

Il secondo algoritmo, di cui si è fatta menzione precedentemente, fonda le basi per lo sviluppo di algoritmi più sofisticati che sono l'oggetto di analisi di questa tesi.
L'algoritmo di Grover fu presentato dal suo inventore nel 1996 [10]. Grazie allo sviluppo delle tecnologie odierne, è stato possibile crearne una prima implementazione su hardware solo vent'anni più tardi, nel 2017.
Per capire qual è lo scopo per cui esso è stato sviluppato, bisogna prima affrontare

nel dettaglio il problema di applicazione che prende banalmente il nome di *The Grover's Problem*. Si consideri un insieme di N elementi, generalmente definito database. Al suo interno, è presente un oggetto particolare, chiamato *winner* e semplicemente indicato con w, che si vuole individuare ma di cui non si conosce la posizione.

Dunque, questo rientra tra i problemi di ricerca. Come tale, è dimostrato che, dato un insieme di cardinalità N, nella computazione classica sono necessarie N interrogazioni del database per risolvere il problema nel caso peggiore. L'algoritmo classico opera tramite interrogazioni a forza bruta sul database, in quanto non si possiede alcuna informazione sull'ordinamento interno degli oggetti. Ne consegue che il numero medio di operazioni sufficienti equivale a $\frac{N}{2}$. Quindi, il costo computazionale di risoluzione è $\mathcal{O}(N)$.

Per risolvere questo problema di ricerca, Grover sviluppò un algoritmo quantistico che è di natura probabilistica. Infatti, esso investe di un ruolo di centrale importanza la distribuzione di probabilità relativa alla misurazione del vettore di stato del sistema: dalle entrate di questo oggetto è possibile ricavare la possibilità a ogni istante di essere in un ingresso del database tra tutti quelli possibili. Pertanto, il procedimento di codifica che egli utilizza è il seguente: ad ogni stato di misurazione del vettore di sistema corrisponde un determinato ingresso del database, mentre l'elevazione al quadrato del modulo della relativa amplitude corrisponde alla probabilità che l'ingresso contenga il winner.
Per eseguire l'algoritmo su un circuito quantistico, è necessario stabilire uno stato iniziale in superposition che tenga simultaneamente conto dell'informazione di tutti gli elementi del database. Sapendo che ad ogni possibile oggetto del database deve corrispondere un autostato della base computazionale di misurazione, allora

ciascuno di essi è codificato tramite una stringa di bit binari di lunghezza generica n, cioè:

$$w, x \in \{0,1\}^n, \tag{3.7}$$

dove n è il numero di qubit a cui si fa ricorso. Per semplicità, si supponga che valga esattamente

$$N = 2^n, \tag{3.8}$$

quindi il numero di qubits necessario è

$$\log_2 N. \tag{3.9}$$

Ciò significa che lo spazio di Hilbert $\mathcal{H}$ generato dalla base computazionale, che genera lo spazio hilbertiano, ha dimensione N. In questo modo, tutti i vettori della base computazionale sono in relazione biunivoca con uno degli oggetti del database. Lo stato iniziale $|s\rangle$ è dato da uno stato di superposition uniforme, ossia la distribuzione di probabilità della realizzazione in seguito ad una misurazione è uniforme:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \,. \tag{3.10}$$

La distribuzione rispecchia l'informazione iniziale che si possiede sugli elementi totali. Tale stato si ottiene tramite l'applicazione simultanea di n Hadamard gates su tutti i qubits, i quali sono sempre inzializzati a $|0\rangle$. Perciò, in alternativa, si può anche dire che

$$|s\rangle = H^{\bigotimes n} |0\rangle^n \,. \tag{3.11}$$

Lungo i passi dell'algoritmo è importante saper determinare l'identità degli elementi. Ossia, una volta interrogato il database in una delle sue posizioni, bisogna essere in grado di riconoscere se essa è occupata dal winner. Pertanto, è necessario definire una codifica, rappresentata da una funzione black-box $f : \mathbb{Z} \to \{0,1\}$, definita *oracle function*, tale che:

- $f(x) = 0$ per tutti gli oggetti $x \neq w$;
- $f(w) = 1$ in corrispondenza del winner.

Associata a questa funzione, è definita un operatore oracle il cui design su circuito è implementato tramite una sequenza opportuna di gates. In generale, è rappresentato da una matrice unitaria U_f, che agisce così:

$$U_f |x\rangle = (-1)^{f(x)} |x\rangle . \tag{3.12}$$

Ne consegue che

$$U_f |x\rangle = \begin{cases} |x\rangle & \text{se } x = w \\ -|x\rangle & \text{altrimenti.} \end{cases}$$

Geometricamente, U_f opera un ribaltamento del vettore $|w\rangle$ rispetto all'origine nello spazio n dimensionale in cui è immerso.

Definizione 3.2.1. *Una trasformazione Householder è una trasformazione lineare di un vettore in cui viene eseguita la sua riflessione rispetto ad un iperpiano contenente l'origine.*

U_f è una trasformazione Householder. Le proprietà di tale operatore coincidono con quelle di una matrice unitaria hermitiana.
Il particolare design dell'operatore oracle su circuito è la seguente: esso agisce su due qubits:

$$U_f : |x\rangle |q\rangle \rightarrow |x\rangle |q \oplus f(x)\rangle , \tag{3.13}$$

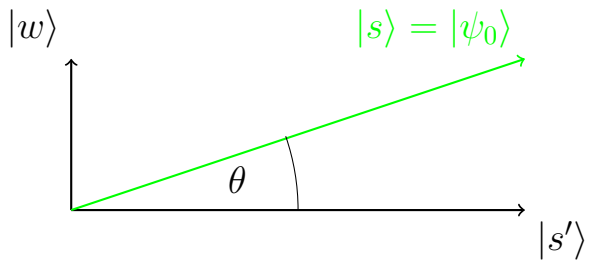
dove:

- $\oplus$ è l'operatore di somma nell'anello della classe di resto 2;
- $|x\rangle$ rappresenta l'indice binario dell'elemento;
- $|q\rangle$ è chiamato *oracle qubit*. Esso conserva l'output dell'operatore ed è inizializzato allo stato di superposition $\frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$.

3.3 Amplitude Amplification

Lo step peculiare dell'algoritmo di Grover, che rende il Quantum Computing computazionalmente più efficace rispetto alla risoluzione di stampo classico, è il

cosiddetto *Amplitude Amplification*. Questa procedura, di natura iterativa, esegue un trasporto di massa di probabilità in modo tale che l'amplitude relativa allo stato del winner sia man mano aumentata a discapito delle altre realizzazioni. Si ricorda che i moduli delle amplitudes corrispondono alla radice quadrata delle probabilità di realizzazione degli autostati in seguito ad una misurazione rispetto alla base computazionale. Quindi, essendo la probabilità di ciascun autostato direttamente proporzionale al modulo della sua amplitude, allora Amplitude Amplification permette di aumentare la probabilità di misurazione dello stato winner.
Per capire qual è l'effetto della procedura, si prenda in considerazione lo spazio vettoriale N dimensionale sul campo dei complessi $\mathbb{C}^N$, che è generato da $span\{|w\rangle, |s'\rangle\}$. Il vettore $|s'\rangle$ è ottenuto rimuovendo dallo stato di superposition uniforme l'amplitude relativa all'elemento winner e, successivamente, viene riscalata la distribuzione dello stato in modo che soddisfi la condizione di normalizzazione. In questo modo $|w\rangle$ e $|s'\rangle$ risultano ortogonali.



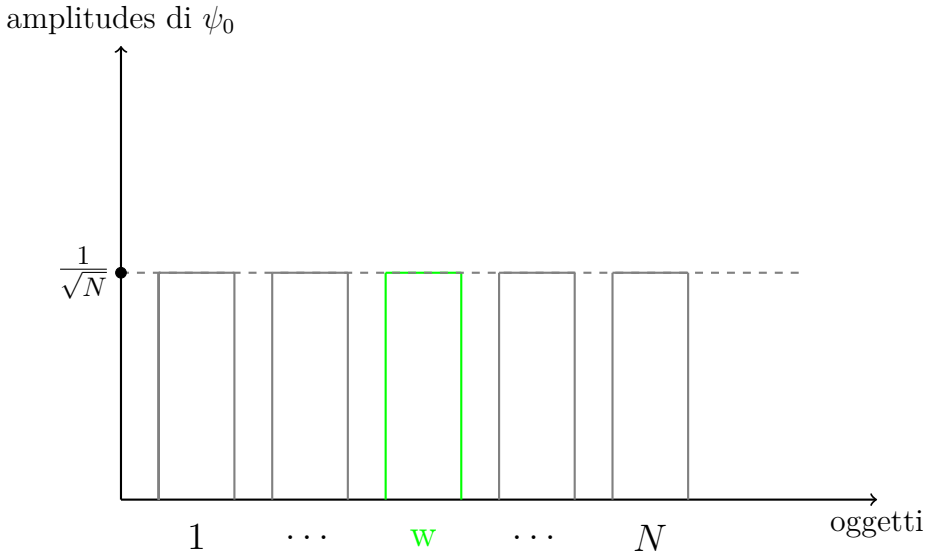
Dal grafico rappresentato nel piano cartesiano, si può osservare che $|s\rangle$ si esprime anche in funzione dell'angolo θ formato da $|s\rangle$ e $|s'\rangle$:

$$|s\rangle = \sin(\theta)\,|w\rangle + \cos(\theta)\,|s'\rangle\,, \tag{3.14}$$

dove

$$\theta = \arcsin(\langle s|w\rangle) = \arcsin\left(\frac{1}{\sqrt{N}}\right), \quad \theta \in [0, \frac{\pi}{2}]. \tag{3.15}$$

Le amplitudes dello stato inziale $|\psi_0\rangle$, o $|s\rangle$, sono rappresentabili con un grafico a barre. Esse possiedono la stessa altezza, equivalente a $\frac{1}{\sqrt{N}}$, che è anche la media dei loro valori.



Applicando la funzione oracle U_f allo stato del sistema $|\psi_0\rangle$ si ottiene un nuovo vettore ribaltato rispetto all'asse $|s'\rangle$, cioè $|\psi_1\rangle$:

$$|\psi_1\rangle = U_f\,|\psi_0\rangle = U_f\,|s\rangle\,. \tag{3.16}$$

Pertanto, tenendo conto della riflessione del vettore, l'operatore U_f può essere espresso anche come

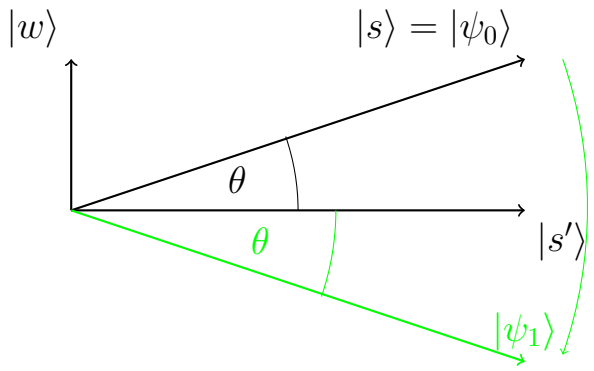
$$U_f = I - 2\,|w\rangle\,\langle w|\,. \tag{3.17}$$

Il prodotto vettoriale $|w\rangle\,\langle w|$ è una matrice P di dimensione $N \times N$. Essendo rispettata la condizione di idempotenza $P^2 = P$, essa applica al vettore di stato una trasformazione di proiezione $P : \mathcal{H} \to \mathcal{H}$. Ciò prova che U_f è un operatore hermitiano. P viene utilizzato per dividere lo spazio $\mathcal{H}$ in due sottospazi:

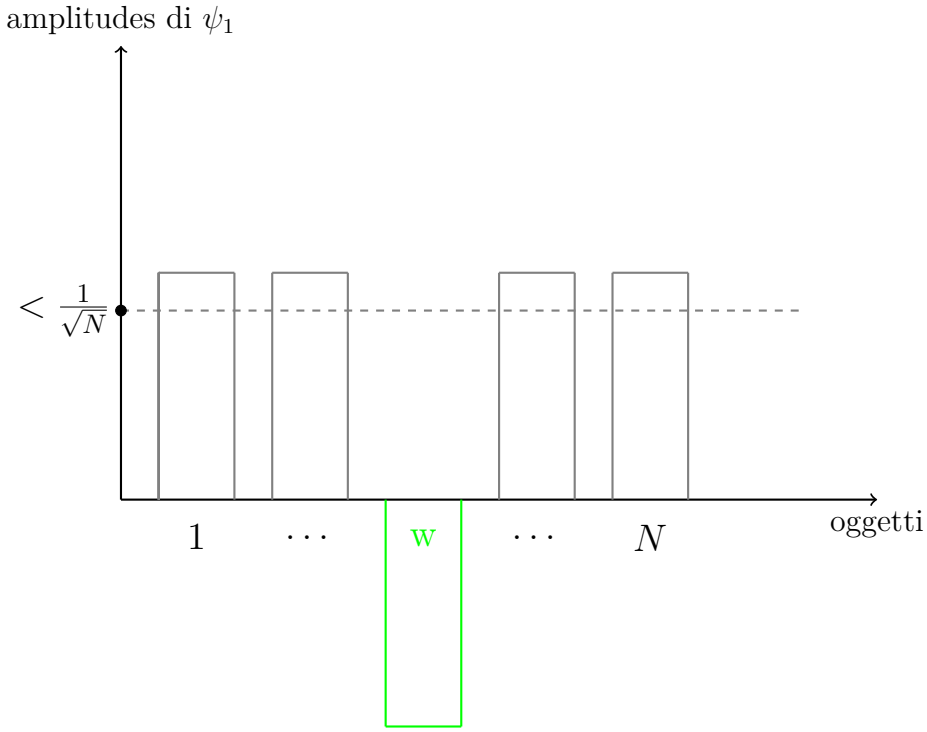
- $\mathcal{H}_0$, detto anche *bad subspace*, costituisce il nucleo dell'operatore di proiezione a cui appartengono tutti gli autostati x della base computazionale tali per cui $f(x) = 0$;
- $\mathcal{H}_1$, oppure *good subspace*, costituisce la direzione dell'operatore di proiezione in cui giace l'autostato w tale che $f(w) = 1$.

Inizialmente, il vettore di stato $|s\rangle$ appartiene a $\mathcal{H} \setminus \mathcal{H}_1$. Tramite il processo iterativo di amplitude amplification, si cerca di proiettare tale vettore sul sottospazio

$\mathcal{H}_1$. L'algoritmo raggiunge il suo scopo quando il modulo di questa proiezione è più grande possibile. A quel punto, la distribuzione di probabilità che è codificata all'interno delle amplitudes del vettore di stato conduce all'ingresso dello stato winner.
L'applicazione di U_f al vettore $|s\rangle$ provoca la seguente riflessione:



L'amplitude che fa riferimento all'autostato $|w\rangle$ è diventata di segno negativo. Indi per cui, l'ampiezza media diminuisce di una certa quantità rispetto al passo precedente:



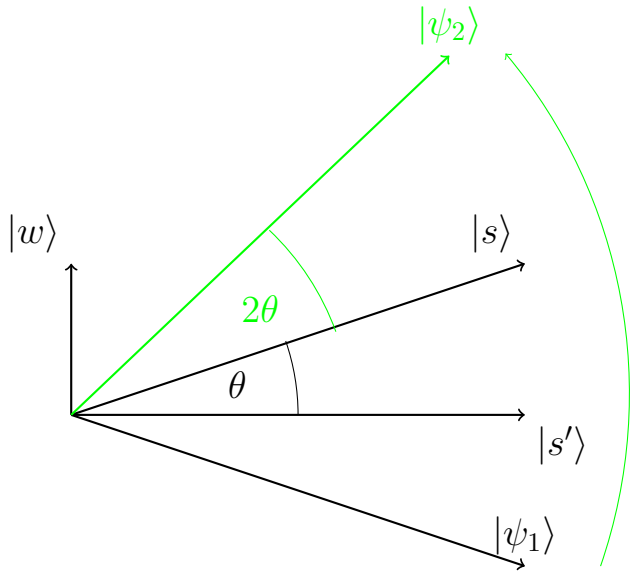
Dopodiché, come ultima operazione dell'iterazione, si applica un ulteriore operatore U_s, anch'esso Householder:

$$U_s = 2 \left|s\right\rangle \left\langle s\right| - I. \tag{3.18}$$

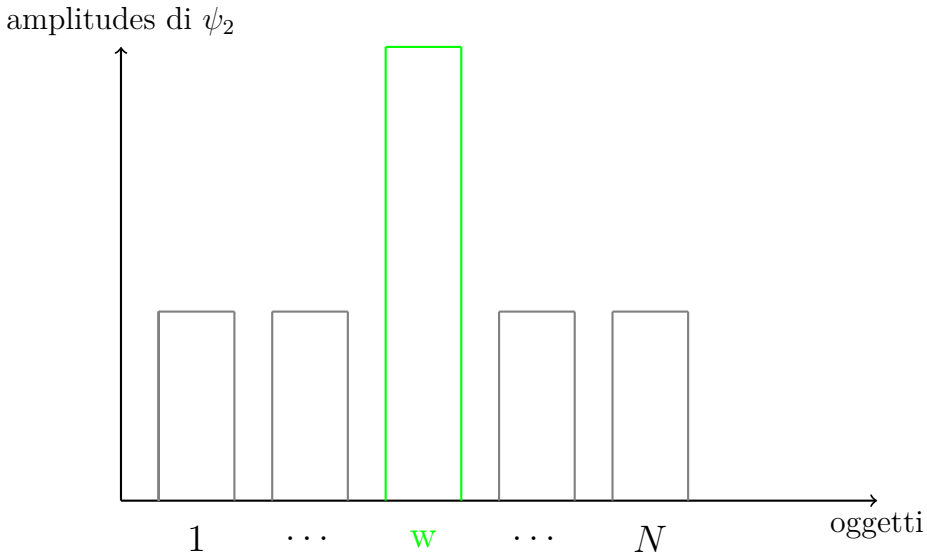
U_s comporta una seconda rotazione rispetto all'asse individuato da $|s'\rangle$, rendendo di nuovo positivi tutti i segni delle amplitudes del vettore di stato:

$$\left|\psi_2\right\rangle = U_s \left|\psi_1\right\rangle = (2 \left|s\right\rangle \left\langle s\right| - I) \left|\psi_1\right\rangle = U_s U_f \left|s'\right\rangle . \tag{3.19}$$

Raffigurando nuovamente su un piano cartesiano, l'operazione avviene come segue:



L'operatore U_s è definito *reflection about the average*: questo nome è indotto dal risultato che esso produce sulle amplitudes del vettore di stato nel grafico a barre riportato di seguito. Infatti, rispetto alla media precedente, la probabilità di misurazione del winner viene riflessa, a discapito di quelle che si riferiscono a tutti gli altri oggetti del database:



La matrice U_s è costruita nel seguente modo:

$$U_{s\{i,j\}} = \begin{cases} \frac{2}{N} & \text{se } i \neq j \\ \frac{2}{N} - 1 & \text{se } i = j, \end{cases}$$

dove $i, j = 1, \ldots, N = 2^n$.
A livello circuitale, U_s è il risultato della seguente combinazione di operatori Hadamard H e di rotazione R:

$$U_s = HRH, \tag{3.20}$$

dove

$$H = 2^{-\frac{n}{2}}(-1)^{i_2 \cdot j_2} \quad R_{ij} = \begin{cases} 0 & \text{se } i \neq j \\ 1 & \text{se } i = 0 \text{ e } i = j \\ -1 & \text{se } i = 0 \text{ e } i = j. \end{cases}$$

e i_2, j_2 sono la rappresentazione binaria degli indici i, j, rispettivamente.

Quindi, un'iterazione di amplitude amplification è resa possibile tramite l'operazione

$$(U_s U_f) \left|s\right\rangle \tag{3.21}$$

e, una volta compiuta, la probabilità di misurazione del winner aumenta da

$$\left|\langle w|s\rangle\right|^2 = \frac{1}{N} \tag{3.22}$$

a

$$\left|\langle w|U_s U_f|s\rangle\right|^2 = \left|\frac{1}{N} \cdot \frac{N-4}{N} + \frac{2}{\sqrt{N}}\right|^2 = \frac{9}{N} \cdot \left(1 - \frac{4}{3N}\right)^2. \tag{3.23}$$

Questo si può dimostrare, facendo osservare quello che accade durante la pima applicazione di amplitude amplification:

$$U_f \left|s\right\rangle = (I - 2\left|w\right\rangle \left\langle w\right|) \left|s\right\rangle = \left|s\right\rangle - 2\left|w\right\rangle \langle w|s\rangle = \left|s\right\rangle - \frac{2}{\sqrt{N}} \left|w\right\rangle, \tag{3.24}$$

$$U_s\left(|s\rangle - \frac{2}{\sqrt{N}}|w\rangle\right) = (2|s\rangle\langle s| - I)\left(|s\rangle - \frac{2}{\sqrt{N}}|w\rangle\right)$$
$$= 2|s\rangle\langle s|s\rangle - |s\rangle\frac{4}{\sqrt{N}}|s\rangle\langle s|w\rangle + \frac{2}{\sqrt{N}}|w\rangle$$
$$= 2|s\rangle - |s\rangle - \frac{4}{\sqrt{N}}\cdot\frac{1}{\sqrt{N}}|s\rangle + \frac{2}{\sqrt{N}}|w\rangle$$
$$= \frac{N-4}{N}|s\rangle + \frac{2}{\sqrt{N}}|w\rangle\,. \quad (3.25)$$

$$p(w) = \left|\langle\frac{N-4}{N}|s\rangle + \frac{2}{\sqrt{N}}|w\rangle\,|w\rangle\right|^2$$
$$= \left|\langle\frac{N-4}{N}|s\rangle\,|w\rangle + \langle\frac{2}{\sqrt{N}}|w\rangle\,|w\rangle\right|^2$$
$$= \left|\frac{1}{N}\cdot\frac{N-4}{N} + \frac{2}{\sqrt{N}}\right|^2. \quad (3.26)$$

Pertnato, l'esecuzione di t iterazioni di Amplitude Amplification, portano allo stato finale

$$|\psi\rangle = (U_s U_f)^t|s\rangle\,. \quad (3.27)$$

3.4 I passi dell'algoritmo di Grover ed il quantum speed-up

Sono stati fin qui introdotti tutti gli elementi che agiscono all'interno dell'algoritmo. Di seguito, viene fornita la successione degli steps dell'Algoritmo di Grover:

1. **inizializzazione**: viene inizializzato il vettore di stato iniziale ψ_0:

$$|\psi_0\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle\,; \quad (3.28)$$

2. **amplitude amplification**: si applica l'operatore di diffusione al vettore di stato della $i-1$-esima iterazione:

$$|\psi_i\rangle = U_s U_f|\psi_{i-1}\rangle\,; \quad (3.29)$$

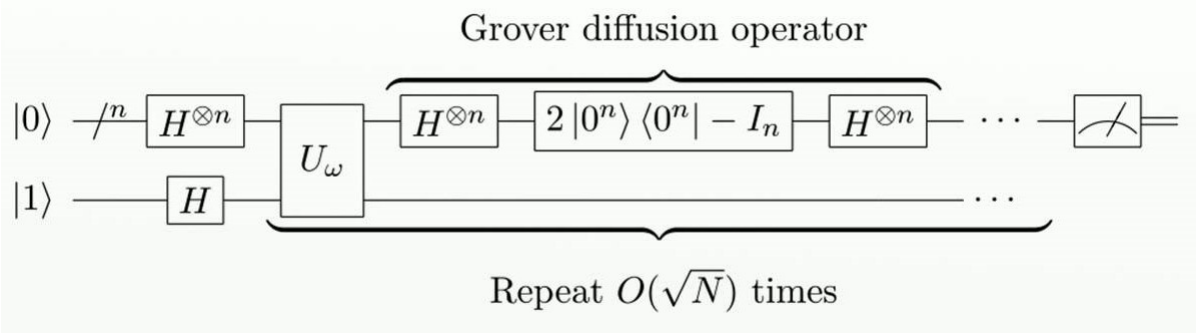


Figura 3.1: L'algoritmo di Grover rappresentato nella sua forma più compatta e generica su un circuito quantistico in cui vengono sfruttati n qubit e il risultato del measurement viene importato su un bit classico.

3. **iterazioni**: si ripete il passo 2 un numero $\mathcal{O}(\sqrt{N})$ di volte;

4. **measurement**: si esegue una misurazione sul vettore di stato che permette di fare un campionamento dalla distribuzione di probabilità finale e definire l'output.

Il terzo step stabilisce di eseguire un numero $\mathcal{O}(\sqrt{N})$ di iterazioni di amplitude amplification. Questo garantisce che l'algoritmo raggiunga la soluzione sperata *w.h.p.*.

Definizione 3.4.1. *Si dice che un algoritmo termina w.h.p. (with high probability) in un tempo $\mathcal{O}(N)$ se ciò avviene con probabilità almeno uguale a $1 - \frac{1}{N^c}$ per ogni $c \geq 1$. [11]*

Dunque, Grover dimostra che il problema da lui esposto può essere risolto in $\mathcal{O}(\sqrt{N})$ con probabilità limitata inferiormente da $1 - \sqrt{\frac{1}{N^c}}$. La dimostrazione di questa affermazione viene ampiamente illustrata nel seguito.
Questo algoritmo rientra tra i problemi di Computer Science che vengono definiti *BQP*, ossia *bounded-error quantum polynomial time*.

Definizione 3.4.2. *Secondo la teoria computazionale quantistica, BQP è una classe di problemi di decisione che sono risolti da un algoritmo quantistico in un tempo polinomiale con un errore probabilistico limitato da $\frac{1}{3}$.*

Il limite di errore $\frac{1}{3}$, in realtà, è arbitrario. Infatti, essendo i problemi di ricerca risolti da un algoritmo quantistico probabilistico, è sufficiente eseguire un numero

precisato di iterazioni tale per cui tale limite sia rispettato. Esso coincide con il complemento dell'errore riportato nella definizione (3.4.1).

L'equazione (3.16) fornisce la decomposizione dello stato iniziale nello spazio generato da $\psi = span(|w\rangle, |s'\rangle)$. La probabilità di misurare lo stato winner corrisponde a $\sin^2(\theta)$. Dato che:

$$U_s U_f |s'\rangle = (2\cos^2(\theta) - 1)|s'\rangle + 2\sin(\theta)\cos(\theta)|w\rangle = \cos(2\theta)|s'\rangle + sin(2\theta)|w\rangle \tag{3.30}$$

$$U_s U_f |w\rangle = -2\sin(\theta)\cos(\theta)|s'\rangle + (1 - 2\sin^2(\theta))|w\rangle = -sin(2\theta)|s'\rangle + \cos(2\theta)|w\rangle, \tag{3.31}$$

segue che un'iterazione di amplitude amplification corrisponde alla rotazione del vettore di stato di un angolo 2θ in ψ.
L'esecuzione di t iterazioni di amplitude amplification corrisponde a:

$$(U_s U_f)^t |\psi_0\rangle = \cos((2t+1)\theta)|s'\rangle + \sin((2t+1)\theta)|w\rangle. \tag{3.32}$$

La probabilità di misurare lo stato winner, a questo punto, corrisponde a

$$p(w) = \sin^2((2t+1)\theta). \tag{3.33}$$

In generale, dato un angolo α, vale la formula di duplicazione:

$$2\sin^2\alpha = 1 - \cos 2\alpha. \tag{3.34}$$

Applicando (3.34) in (3.33), allora la probabilità di misurazione è data da

$$p(w) = \frac{1 - \cos(2(2t+1)\theta)}{2}, \tag{3.35}$$

che raggiunge il suo valore massimo in corrispondenza di un numero di iterazioni uguale a

$$t = \lfloor \frac{\pi}{4\theta} \rfloor. \tag{3.36}$$

Se $|\sin\theta| \ll 1$, si può compiere la seguente approssimazione:

$$\theta \backsimeq \sin\theta. \tag{3.37}$$

In questo modo, applicando (3.37) alla (3.36):

$$t = \lfloor \frac{\pi}{4\theta} \rfloor = \lfloor \frac{\pi}{4\sin\theta} \rfloor = \lfloor \frac{\pi\sqrt{N}}{4} \rfloor, \tag{3.38}$$

dove, nell'ultima equazione, $\sin\theta = \frac{1}{\sqrt{N}}$, per lo stato iniziale di superposition. Da questo risultato si ricava che

$$t = \lfloor \frac{\pi\sqrt{N}}{4} \rfloor = \mathcal{O}(\sqrt{N}). \tag{3.39}$$

In relazione ad un corrispettivo algoritmo classico di risoluzione, che è in grado di risolvere il problema in un tempo $\mathcal{O}(N)$, viene chiarificato il motivo per cui si parla di *quantum speed-up*. In altri termini, l'algoritmo di Grover gode di un'efficenza computazionale superiore rispetto ai corrispettivi metodi di risoluzione classici. Questa affermazione è scaturita da una dimostrazione di natura teorica, e ci si aspetta che l'hardware quantistico, una volta sufficientemente pronto, ne fornisca anche una diretta evidenza empirica.

Nelle loro ricerche, Viamontes, Markov e Hayes hanno analizzato i punti di forza dell'algoritmo di Grover in [12], ma anche i suoi molteplici impedimenti pratici. Ci sono alcuni svantaggi che sfavoriscono il percorrimento della strada di soluzione fornita dal metodo quantistico. Uno di essi riguarda l'operatore oracle. E' vero che, essendo una routine computazionale, per convenzione si assume che esso comporti un costo dell'ordine di $\mathcal{O}(1)$. Tuttavia tale funzione richiede un'esplicita implementazione sull'hardware, che molto spesso esige un gravoso lavoro da parte degli sviluppatori. Inoltre, essa varia in relazione al problema a cui viene applicata. Pertanto, la mole di lavoro che l'implementazione ad-hoc di una funzione richiede per un preciso problema di Grover, in generale, non può essere spesa anche per un problema diverso.

Un secondo punto negativo riguarda la scalabilità dell'oracle: usualmente, la successione dei gates, che ne costituiscono il corpo di implementazione sull'hardware, ha una lunghezza proporzionale alla dimensione del problema. Se tale grandezza

è elevata, allora i circuiti quantistici che ne simulano il comportamento risultano eccessivamente profondi. Di conseguenza, gli errori di decoerenza, tipici della tecnologia quantistica, provocano un errore di distorsione del risultato, a tal punto da non renderlo trascurabile.
Tuttavia, anche a fronte di queste complicazioni, l'esigenza che ha portato Grover a formulare questo algoritmo di ricerca, non riguarda l'ottenimento del risultato. Infatti, gli algoritmi classici sono in grado di fornire un risultato corretto senza che venga scomodata la fisica quantistica. Piuttusto, esso funge da esperimento teorico per fornire la prova che dimostrerebbe la supremazia computazionale del QC in termini di efficienza, ristretto a particolari contesti matematici.

3.5 Il problema multiplo

L'algoritmo di Grover è stato teorizzato per risolvere un problema di ricerca di un solo elemento all'interno di un database. Quantum Amplitude Amplification è un algoritmo che rientra in uno dei suoi steps, come precedentemente è stato illustrato in questo capitolo. Tuttavia, QAA può essere generalizzato al caso in cui gli elementi winners sono in numero maggiore di uno soltanto. Dunque, si ipotizzi che questi oggetti siano in un numero uguale a $M > 1$ e che N sia la cardinalità de database. Allora, il costo computazionale di esecuzione è $\mathcal{O}(\sqrt{\frac{N}{M}})$ [13]. Ripercorrendo la dimostrazione che è stata utilizzata per il caso più semplice, cioè quello unitario, si indichi di nuovo il vettore inziale in superposition:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \,. \tag{3.40}$$

Nel caso multidimensionale, l'operatore di proiezione $P = |s\rangle \langle s|$ della (3.17), diventa

$$P = \sum_{i=1}^{M} |w_i\rangle \langle w_i| \,, \tag{3.41}$$

dove $\forall i = 1, \ldots, M$ la funzione black-box f valutata in w_i restituisce 1, altrimenti 0.
Come prima, P divide lo spazio di Hilbert $\mathcal{H}$ nei due sottospazi good subspace e bad subspace, dove il primo corrisponde all'immagine di P, in cui giacciono tutti gli

M autostati w_i, con $i = 1, \ldots, M$, della base computazionale, tali che $f(w_i) = 1$. Richiamando θ, ossia l'angolo creato tra $|s'\rangle$ e $|s\rangle$, allora la probabilità che uno degli oggetti venga misurato è data da:

$$\sin^2 \theta = \frac{M}{N}. \tag{3.42}$$

Dunque, utilizzando di nuovo (3.36) e (3.37), vale

$$t = \lfloor \frac{\pi}{4\theta} \rfloor = \lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \rfloor = \mathcal{O}(\sqrt{N}). \tag{3.43}$$

Come si evince da questo risultato, anche nel caso in cui ci siano più elementi winner, è dimostrato che QAA fornisce uno speed-up quadratico rispetto al costo computazionale di un algoritmo di risoluzione classico.

3.6 Quantum Amplitude Estimation

Grover's Algorithm e QAA aprono la strada alla trattazione di un altro notevole algoritmo quantistico. *Quantum Amplitude Estimation* è un algoritmo quantistico di grande avanguardia, perchè permette la stima di quantità statistiche con uno speed-up quadratico computazionale rispetto ai metodi di risoluzioni classici Monte Carlo. Ovviamente, ciò è sostenuto sulla base teorica su cui anche i precedenti algoritmi sono stati esposti. La scoperta di QAA è stata condotta da Gilles Brassard, il quale si impegnò a costruire un'estensione del Grover's Algorithm per un problema più complesso [13].

Sia dato il problema multiplo di Grover: in un database di N elementi in cui il numero totale di oggetti winners è M, ossia

$$M = |\{x \in X | f(x) = 1\}|, \tag{3.44}$$

dove $f : X \to \{0,1\}$ è una funzione booleana black-box.
Finora si è fatto coincidere il numero di elementi N con una potenza di 2, in modo tale che $N = 2^n$, con n numero di qubits. Non sempre è possibile fare ciò, perchè

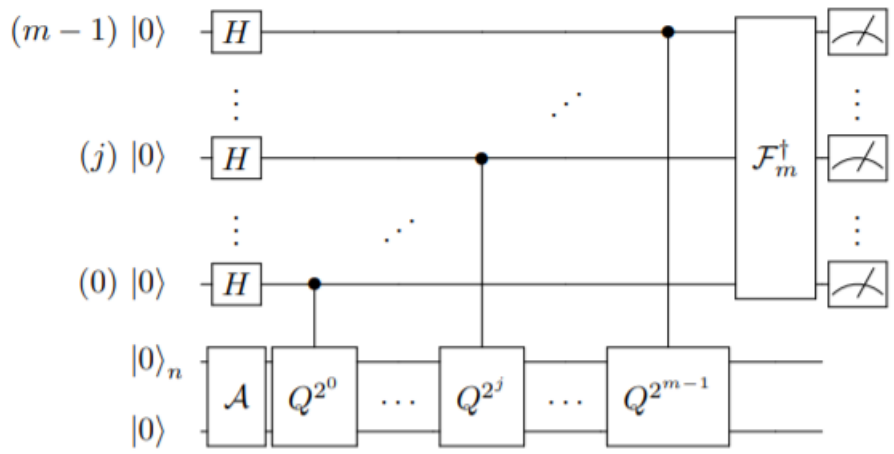


Figura 3.2: Circuito quantistico per QAE.

le ipotesi del problema non lo consentono. Quindi è necessario introdurre una notazione più generale che viene utilizzata per il caso più generico del problema. Il vettore di stato $|\psi\rangle$ può essere espresso tramite la seguente superposition:

$$|\psi\rangle = |\psi_0\rangle + |\psi_1\rangle\,, \tag{3.45}$$

ossia in funzione delle sue proiezioni sui due sottospazi bad $\mathcal{H}_0$ e good $\mathcal{H}_1$ inclusi nello spazio hilbertiano $\mathcal{H}$. Si ricorda che il bad subspace $\mathcal{H}_0$ è l'insieme dei vettori di stato di misurazione rispetto alla base computazionale che sono associati ad un elemento non winner del database. Mentre il good subspace $\mathcal{H}_1$ comprende i vettori di misurazione che codificano gli elementi winners.
L'obbiettivo di QAE corrisponde a trovare la stima di

$$a = \langle\psi_1|\psi_1\rangle\,, \tag{3.46}$$

che è la probabilità di misurare lo stato del sistema in uno stato *good*. Associato a questo problema, coesiste un problema di counting, in quanto non si conosce a priori quale sia il preciso valore intero di M.
Lo spazio spannato dai vettori ψ_0 e ψ_1 ha dimensione 2 se e solo se $0 < a < 1$; e dimensione 1, altrimenti.

Il circuito quantistico che definisce l'algoritmo QAE è rappresentato in Figura 3.2 e nel seguito viene fornita la spiegazione dei passaggi di implementazione.
Si supponga di avere due registri composti rispettivamente da m e $n+1$ qubits: come ogni algoritmo quantistico impone, i qubits iniziali sono sempre inizializzati al valore $|0\rangle$. Il primo registro è anche chiamato *counting register* ed è la sede in cui viene codificato il risultato che si ricava tramite un processo di misurazione; il secondo, di contro, viene utilizzato per poter codificare le informazioni del problema specifico. Tramite le porte quantistiche che agiscono sull'intero circuito, i due registri vengono messi in comunicazione reciproca.
Come primo passo, nel counting register viene applicata un'operazione di trasformazione lineare del sottovettore di stato, che permette di raggiungere uno stato di superposition uniforme. Questa operazione è eseguita tramite l'utilizzo di gates che agiscono in funzione delle sottofasi di ogni qubit del registro. Usualmente, tali sottofasi sono descritte in funzione di una serie ordinata di potenze di 2 che termina in N. In generale, però, non è sempre così. A volte, le sottofasi non coincidono con potenze di 2. Quindi è opportuno definire tale operazione in maniera che valga anche per il caso più generale. Pertanto, è opportuno definire l'operazione comune con cui viene raggiunto lo stato di superposition uniforme: *Quantum Fourier Transform.*

Definizione 3.6.1. *QFT è un algoritmo quantistico notevole, che esegue una trasformazione lineare F_N da uno stato quantistico*

$$|x\rangle = |x_0 x_1 \dots x_{N-1}\rangle = \sum_{j=0}^{N-1} x_j \, |j\rangle \tag{3.47}$$

allo stato

$$|y\rangle = |y_0 y_1 \dots y_{N-1}\rangle = \sum_{j=0}^{N-1} y_j \, |j\rangle \, , \tag{3.48}$$

dove x_0 e y_0 sono i bit più significativi. La forma analitica che la descrive è

$$F_N : |x_k\rangle \rightarrow |y_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j w_N^{jk}, \tag{3.49}$$

con $w_N^{jk} = e^{2\pi ijk/N}$.
La matrice unitaria hermitiana associata è

$$F_N = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & \dots & w^{N-1} \\ 1 & w^2 & w^4 & \dots & w^{2(N-1)} \\ 1 & w^3 & w^6 & \dots & w^{3(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & \dots & w^{(N-1)(N-1)} \end{pmatrix}$$

Definito M intero positivo, l'operatore QFT esegue la seguente trasformazione:

$$F_M : |x\rangle \to \frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} e^{2\pi ixy/M} |y\rangle \qquad 0 \leq x < M, \tag{3.50}$$

che, in forma più esplicita, risulta essere:

$$\frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} e^{2\pi ixy/M} |y\rangle = \frac{1}{\sqrt{M}} \bigotimes_{y=1}^{n} \left(|0\rangle + e^{2\pi ixy/M} |1\rangle \right). \tag{3.51}$$

Nel caso in cui M è una potenza di 2, e cioè $M = 2^m$, per un qualche n positivo, allora la (3.51) equivale a

$$\begin{aligned} \frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} e^{2\pi ixy/M} |y\rangle &= \frac{1}{\sqrt{2^m}} \bigotimes_{y=0}^{m} \left(|0\rangle + e^{\frac{2\pi ix}{2^y}} |1\rangle \right) \\ &= \frac{1}{\sqrt{2^m}} \left(|0\rangle + e^{\frac{2\pi ix}{2}} |1\rangle \right) \bigotimes \cdots \bigotimes \left(|0\rangle + e^{\frac{2\pi ix}{2^m}} |1\rangle \right). \end{aligned} \tag{3.52}$$

In tale definizione, quando il numero totale N di stati, associati biunivocamente agli autostati della base computazionale, corrisponde ad una potenza di 2, allora QFT equivale ad applicare una porta di Hadamard per ogni qubit del registro:

$$H^{\otimes m} |0\rangle_m . \tag{3.53}$$

Il secondo registro, composto da $n + 1$ qubits, subisce una inizializzazione che viene impartita con l'operatore $\mathcal{A}$, che contiene le informazioni del problema.

Sull'ultimo qubits, definito *objective qubit*, viene generalmente misurato l'output del problema. Il sottostato del secondo registro risulta

$$\mathcal{A}\left|0\right\rangle_{n+1} = \sqrt{1-a}\left|\psi_0\right\rangle_n\left|0\right\rangle + \sqrt{a}\left|\psi_1\right\rangle_n\left|1\right\rangle, \tag{3.54}$$

in cui $a \in [0,1]$ è l'incognita del problema e rappresenta la probabilità di misurare $|1\rangle$ sull'ultimo qubit. Il problema su hardware quantistico è costruito in modo tale che la misurazione di autostato $|1\rangle$ sia codificato alla statistica di interesse che corrisponde all'obbiettivo del problema.
A questo punto viene applicato un operatore $\mathcal{Q}$ per m volte consecutive. Esso è un operatore di controllo, nel senso che viene applicato al secondo registro solo se il qubit di controllo del primo registro a cui è collegato si trova in stato $|1\rangle$:

$$\mathcal{Q} = -\mathcal{A}S_0\mathcal{A}^{-1}S_{\psi_0}, \tag{3.55}$$

dove:

- $\mathcal{A}$ è un algoritmo quantistico che non fa uso di measurement e che pone inizialmente il vettore di stato in uno di superposition, in cui vengono caricate le informazioni del problema;
- $\mathcal{A}^{-1}$ è un algoritmo quantistico che corrisponde all'operazione inversa del precedente;
- $S_0 = I - 2\left|0\right\rangle\left\langle 0\right|$ inverte il segno delle amplitudes degli stati winners se e solo se lo stato che riceve in input è $|0\rangle$;
- S_{ψ_0} equivale a quello che nell'algoritmo di Grover viene indicato con U_f, cioè l'operatore oracle. S_{ψ_0} rappresenta il ribaltamento rispetto al sottostato $|\psi_0\rangle_n$ dei primi n qubits del registro totale.

Per stimare a si applicano una serie di iterazioni dell'operatore $\mathcal{Q}$ definito dalla $(B.2)$, dove

$$S_{\psi_0} = I - 2\left|\psi_0\right\rangle\left|0\right\rangle\left\langle\psi_0\right|\left\langle 0\right| \tag{3.56}$$

comporta una rotazione di 2θ del vettore di stato all'interno dello spazio bidimensionale generato dalla base $\{|\psi_0\rangle, |\psi_1\rangle\}$.

Le riflessioni ottenute con i due operatori U_s e U_f ((3.18), (3.17)), sono rispettivamente generalizzate con i due operatori:

$$S_0 = I - \frac{2}{1-a} |0\rangle \langle 0| \tag{3.57}$$

e

$$S_{\psi_0} = I - 2 |\psi_0\rangle \langle \psi_0| . \tag{3.58}$$

Fisicamente, l'implementazione su circuito degli operatori di riflessione è riportata in Appendice B.
Siccome $\mathcal{Q}$ è unitario e agisce come proiettore del vettore di stato su $\mathcal{H}_\psi$, segue che $\mathcal{H}_\psi$ possiede una base ortonormale composta da due autovettori di $\mathcal{Q}$:

$$|\psi_+\rangle = \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{a}} |\psi_1\rangle + \frac{1}{\sqrt{1-a}} |\psi_0\rangle \right), \tag{3.59}$$

$$|\psi_-\rangle = \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{a}} |\psi_1\rangle - \frac{1}{\sqrt{1-a}} |\psi_0\rangle \right), \tag{3.60}$$

con rispettivi autovalori:

$$\lambda_+ = e^{+i2\theta}, \tag{3.61}$$

$$\lambda_- = e^{-i2\theta}, \tag{3.62}$$

dove $0 < a < 1$.
Ne segue che il vettore in superposition iniziale si esprime anche in funzione degli autovettori che formano la base:

$$\mathcal{A} |0\rangle = |\psi\rangle = \frac{-i}{\sqrt{2}} \left(e^{i\theta} |\psi_+\rangle - e^{-i\theta} \right) |\psi_-\rangle). \tag{3.63}$$

Ecco che, passando anche attraverso un'analisi dello spettro dell'operatore $\mathcal{Q}$, si conferma quanto detto già dalla (3.32).

Sapendo che $a = \sin^2 \theta$, con $0 \leq \theta \leq \frac{\pi}{2}$, e che si conosce lo spettro di $\mathcal{Q}$ dato da (3.61), (3.62), dunque la stima di a segue direttamente da quella di uno dei due autovalori di $\mathcal{Q}$. La stima di θ, ovvero $\hat{\theta}$, non sempre è corretta, in quanto essa viene

prodotta tramite la discretizzazione dei suoi valori e messa in relazione biunivoca con autostati di misurazione rispetto alla base computazionale del counting register. Pertanto, l'errore presente su $\hat{\theta}$ produce una stima $\hat{a} = \sin^2 \hat{\theta}$ di a.
Sapendo inoltre che t iterazioni di $\mathcal{Q}$ possono essere espresse in funzione dei suoi autovalori, allora vale

$$\begin{aligned}\mathcal{Q}^t |\psi\rangle &= \frac{-i}{\sqrt{2}} \left(e^{(2t+1)i\theta} |\psi_+\rangle - e^{-(2t+1)i\theta} \right) |\psi_-\rangle) \\ &= \frac{1}{\sqrt{a}} \sin\left((2t+1)\theta\right) |\psi_1\rangle + \frac{1}{\sqrt{1-a}} \cos\left((2t+1)\theta\right) |\psi_0\rangle , \quad (3.64)\end{aligned}$$

dove $|\psi_1\rangle$, $|\psi_0\rangle$ sono ancora i vettori, dati dalla (3.45). Come si evince, le amplitudes di misurare uno stato good o bad sono funzioni sinussoidali di periodo $\frac{\pi}{\theta}$.

Dopo aver inizializzato i due registri agli opportuni stati di superposition, si applica al secondo di essi l'operatore $\Lambda(Q)$.

Definizione 3.6.2. *Si definisca l'operatore* Λ*:*

$$\Lambda(U) : |j\rangle |y\rangle \rightarrow |j\rangle \left(U^j |y\rangle \right), \quad (3.65)$$

dove $0 \leq j < M$ *e* U *è un operatore unitario.*
Dato $e^{2\pi i w}$ *appartenente allo spettro di* U *con relativo autovettore* $|\phi\rangle$*, segue direttamente che*

$$\Lambda_M(U) |j\rangle |\phi\rangle = e^{2\pi i w} |j\rangle |\phi\rangle . \quad (3.66)$$

In sostanza, $\Lambda(Q)$ coincide con la serie di m potenze di Q, al fine di stimare la fase del sottovettore generato da $\mathcal{A} |0\rangle_{n+1}$. Ognuna delle m applicazioni di $\mathcal{Q}$ costituisce un *quantum sample* per rappresentare il risultato dell'algoritmo eseguito sul secondo registro.
Si osservi infatti che, detto $|\psi_+\rangle$ l'autovettore di $\mathcal{Q}$ con autovalore $e^{2\pi i\theta}$ [14]:

$$\mathcal{Q}^{2^m} |\psi_+\rangle = \mathcal{Q}^{2^m-1} \mathcal{Q} |\psi_+\rangle = \mathcal{Q}^{2^m-1} e^{2\pi i\theta} |\psi_+\rangle = \cdots = e^{2\pi i 2^m \theta} |\psi_+\rangle . \quad (3.67)$$

L'applicazione di $\mathcal{Q}^{2^0}$ al m-esimo qubit del primo registro equivale al seguente vettore di stato:

$$\begin{aligned}
&\frac{1}{2^{\frac{1}{2}}}\left(|0\rangle|\psi\rangle + |1\rangle e^{2\pi i 2^0\theta}|\psi\rangle\right)\bigotimes_{k=1}^{m-1}\frac{1}{2^{\frac{m-1}{2}}}\left(|0\rangle+|1\rangle\right)\\
&\qquad=\frac{1}{2^{\frac{1}{2}}}\left(|0\rangle|\psi\rangle + e^{2\pi i 2^0\theta}|1\rangle|\psi\rangle\right)\bigotimes_{k=1}^{m-1}\frac{1}{2^{\frac{m-1}{2}}}\left(|0\rangle+|1\rangle\right)\\
&\qquad=\frac{1}{2^{\frac{1}{2}}}\left(|0\rangle + e^{2\pi i 2^0\theta}|1\rangle\right)|\psi\rangle\bigotimes_{k=1}^{m-1}\frac{1}{2^{\frac{m-1}{2}}}\left(|0\rangle+|1\rangle\right)\\
&\qquad=\frac{1}{2^{\frac{m}{2}}}\left(|0\rangle + e^{2\pi i 2^0\theta}|1\rangle\right)\bigotimes_{k=1}^{m-1}\left(|0\rangle+|1\rangle\right)|\psi\rangle\,, \quad (3.68)
\end{aligned}$$

dove $|0\rangle|\psi\rangle + |1\rangle\otimes e^{2\pi i 2^j\theta}|\psi\rangle = \left(|0\rangle + e^{2\pi i 2^j\theta}|1\rangle\right)|\psi\rangle \quad \forall j$.

Applicando l'operatore $\Lambda(\mathcal{Q})$, ossia tutti i quantum sample $\mathcal{Q}^{2^j}$ con $j=0,\ldots,m$, si ottiene il seguente sottostato finale per il primo registro:

$$\frac{1}{2^{\frac{m}{2}}}\sum_{k=0}^{2^m-1}e^{2\pi i k\theta}|k\rangle = \frac{1}{2^{\frac{m}{2}}}\left(|0\rangle + e^{2\pi i 2^{m-1}\theta}|1\rangle\right)\otimes\cdots\otimes\left(|0\rangle+e^{2\pi i 2^0\theta}|1\rangle\right), \quad (3.69)$$

dove $|k\rangle$ indica il k-esimo tra gli autostati della base computazionale. Contemporaneamente, il secondo registro rimane invariato allo stato $|\psi\rangle = \mathcal{A}|0\rangle$.

Definizione 3.6.3. *Per ogni intero $M>0$ e $w\in[0,1)$*

$$|S_M(w)\rangle = \frac{1}{\sqrt{M}}\sum_{y=0}^{M-1}e^{2\pi i w y}|y\rangle\,. \quad (3.70)$$

Si osservi che la (3.69) equivale alla (3.70).
Dalla precedente definizione, consegue che $|S_M(w)\rangle$ codifica l'angolo $2\pi w$ nelle fasi degli M autostati misurati rispetto alla base computazionale a partire da un vettore in superposition uniforme. Infatti,

$$F_M|x\rangle = |S_M\left(\frac{x}{M}\right)\rangle\,. \quad (3.71)$$

Definizione 3.6.4. *Per ogni coppia di reali w_0 e w_1, sia*

$$d(w_0,w_1) = \min_{z\in\mathbb{Z}}\{|z+w_1-w_0|\}. \quad (3.72)$$

Dunque, la lunghezza di arco minimo, appartenente alla circonferenza goniometrica, che misura la distanza tra $e^{2\pi i w_0}$ e $e^{2\pi i w_1}$ è data da $2\pi d(w_0, w_1)$.

Lemma 3.6.1. *Detto $\Delta = d(w_0, w_1)$ con $w_0, w_1 \in [0,1)$, se $\Delta = 0$, implica che*

$$|\langle S_M(w_0)|S_M(w_1)\rangle|^2 = 1. \tag{3.73}$$

Altrimenti,

$$|\langle S_M(w_0)|S_M(w_1)\rangle|^2 = \frac{\sin^2(M\Delta\pi)}{M^2\sin^2(\Delta\pi)}. \tag{3.74}$$

Dimostrazione.

$$\begin{aligned} |\langle S_M(w_0)|S_M(w_1)\rangle|^2 == & \left|\left(\frac{1}{\sqrt{M}}\sum_{y=0}^{M-1} e^{-2\pi i w_0 y}\langle y|\right)\left(\frac{1}{\sqrt{M}}\sum_{y=0}^{M-1} e^{2\pi i w_1 y}|y\rangle\right)\right|^2 \\ & = \frac{1}{M^2}\left|\sum_{y=0}^{M-1} e^{2\pi i \Delta y}\right|^2 = \frac{\sin^2(M\Delta\pi)}{M^2\sin^2(\Delta\pi)}. \end{aligned} \tag{3.75}$$

□

A questo punto, al primo registro viene applicato *inverse QFT*, ossia l'algoritmo inverso di QFT. In questo modo, si vuole stimare la fase dello stato del counting register dato dalla (3.69), ossia θ.

$$\begin{aligned} F^{-1}\left(\frac{1}{2^{\frac{m}{2}}}\sum_{k=0}^{2^m-1} e^{2\pi i k\theta}|k\rangle\right) &= \frac{1}{2^{\frac{m}{2}}}\sum_{k=0}^{2^m-1} e^{2\pi i k\theta}\frac{1}{2^{\frac{m}{2}}}\sum_{x=0}^{2^m-1} e^{\frac{-2\pi i k x}{2^m}}|x\rangle \\ &= \frac{1}{2^m}\sum_{x=0}^{2^m-1}\sum_{k=0}^{2^m-1} e^{2\pi i k\theta} e^{\frac{-2\pi i k x}{2^m}}|x\rangle \\ &= \frac{1}{2^m}\sum_{x=0}^{2^m-1}\sum_{k=0}^{2^m-1} e^{-\frac{2\pi i k}{2^m}(x-2^m\theta)}|x\rangle. \end{aligned} \tag{3.76}$$

In generale, il problema di stimare $0 \leq w < 1$, dato lo stato finale $|S_M(w)\rangle$ e $w = \frac{x}{M}$ con $0 \leq x < M$, corrisponde al problema inverso di QFT. In altre parole,

si tratta di creare un operatore F_M^{-1} che stimi la fase dello stato passato in input, tale per cui vale:

$$F_M^{-1} \left| S_M \left(\frac{x}{M} \right) \right\rangle = |x\rangle \,. \tag{3.77}$$

Qualora $x = Mw$ non fosse un intero, allora la misurazione di $F_M^{-1} |S_M(w)\rangle$ produce una stima di w. Invece, se $x = Mw$ è un intero, allora la misurazione produce il risultato esatto, ossia con probabilità uguale ad 1. La correttezza di tale valore è descritta dal seguente teorema.

Teorema 3.6.1. *Sia misurato $F_M^{-1} |S_M(w)\rangle$ nella base computazionale e il risultato che ne deriva sia rappresentato da una variabile discreta X.*

1. *se $x = Mw$ è un intero, allora $\mathbb{P}(X = Mw) = 1$.;*

2. *se $x = Mw$ non è un intero:*

$$\mathbb{P}(X = x) = \frac{\sin^2(M\Delta\pi)}{M^2 \sin^2(\Delta\pi)} \leq \frac{1}{(2M\Delta)^2}, \tag{3.78}$$

con $\Delta = d\left(w, \frac{x}{M}\right)$.
Per ogni $k > 1$,

$$\mathbb{P}\left(d\left(w, \frac{X}{M}\right) \leq \frac{k}{M} \right) \geq 1 - \frac{1}{2(k-1)}. \tag{3.79}$$

Nel caso particolare in cui $k = 1$ e $M > 2$,

$$\mathbb{P}\left(d\left(w, \frac{X}{M}\right) \leq \frac{1}{M} \right) \geq \frac{8}{\pi^2}. \tag{3.80}$$

Dimostrazione. 1. Usando il Lemma (3.6.1) nell'ultima equazione, si ottiene:

$$\begin{aligned} \mathbb{P}(X = x) &= \left| \langle x | F^{-1} | S_M(w) \rangle \right|^2 \\ &= \left| (F|x\rangle)^\dagger |S_M(w)\rangle \right|^2 \\ &= \left| \langle S_M\left(\frac{x}{M}\right) | S_M(w) \rangle \right|^2 = 1; \end{aligned} \tag{3.81}$$

2.

$$\mathbb{P}\left(d\left(w,\frac{X}{M}\right)\leq\frac{k}{M}\right)=1-\mathbb{P}\left(d\left(w,\frac{X}{M}\right)>\frac{k}{M}\right)$$
$$\geq 1-2\sum_{j=k}^{\infty}\frac{1}{4M^2\left(\frac{j}{M}\right)^2}$$
$$\geq 1-\frac{1}{2(k-1)}. \quad (3.82)$$

Per $M>2$ si definisca $\Delta=\frac{1}{2M}$ con $0\leq M\leq\frac{1}{M}$, consegue:

$$\mathbb{P}\left(d\left(w,\frac{X}{M}\right)\leq\frac{1}{M}\right)=\mathbb{P}\left(X=\lfloor Mw\rfloor\right)+\mathbb{P}\left(X=\lceil Mw\rceil\right)$$
$$=\frac{\sin^2\left(M\Delta\pi\right)}{M^2\sin^2\left(\Delta\pi\right)}+\frac{\sin^2\left(M\left(\frac{1}{M}-\Delta\right)\pi\right)}{M^2\sin^2\left(\left(\frac{1}{M}-\Delta\right)\pi\right)}\geq\frac{8}{\pi^2}. \quad (3.83)$$

□

Come ultimo passaggio, si compie una misurazione del counting register. Come risultato si ottiene uno degli stati che formano la base computazionale, ossia un vettore binario ad m cifre. Esso coincide alla stima dell'angolo di fase dello stato dato dalla (3.76). Come si può vedere, esso ha un picco per $x=M\theta=2^m\theta$. Seguendo il Teorema (3.6.1), se $x=M\theta$ è un intero, allora la misurazione fornisce il risultato esatto WHP; altrimenti, esso viene arrotondato all'intero più vicino compiendo un errore di approssimazione. Infatti, in questo secondo caso, pur ottenendo il risultato sperato WHP, avviene un procedimento di codifica del punto di massimo attraverso un valore dell'intorno corretto.
Lo stato che si presenta dopo la misurazione è dato da

$$|y\rangle=|2^m\theta\rangle_m\in\{0,1\}^m. \quad (3.84)$$

Dopodiché, si esegue una procedura di calcolo classica, che mappa questo risultato nell'angolo θ cercato:

1. si trasforma $|y\rangle$ dalla notazione binaria in valore intero decimale y_{10} e lo si moltiplica per π;

2. si divide il risultato per la cardinalità di codifica $M = 2^m$, ottenendo

$$\theta = \frac{\pi y_{10}}{2^m}. \tag{3.85}$$

Quanto descritto finora, è supportato dai teoremi seguenti, ampiamente illustrati in [13].

Lemma 3.6.2. *Siano* $a = \sin\theta$, $\hat{a} = \sin\hat{\theta}$, $0 \leq \theta, \hat{\theta} \leq 2\pi$, *allora*

$$\left|\hat{\theta} - \theta\right| \leq \epsilon \Rightarrow |\hat{a} - a| \leq 2\epsilon\sqrt{a(1-a)} + \epsilon^2. \tag{3.86}$$

Dimostrazione. Sia $\epsilon > 0$, l'equazione segue dalla somma membro a membro delle seguenti equazioni trigonometriche:

$$\sin^2\theta + \epsilon - \sin^2\theta = \sqrt{a(1-a)}\sin 2\epsilon + (1-2a)\sin^2\epsilon, \tag{3.87}$$

$$\sin^2\theta - \sin^2\theta - \epsilon = \sqrt{a(1-a)}\sin 2\epsilon + (2a-1)\sin^2\epsilon. \tag{3.88}$$

□

Teorema 3.6.2. *Dato* $k \in \mathbb{Z}_+$, *QAE fornisce il risultato* $0 \leq \hat{a} \leq 1$, *tale che*

$$|\hat{a} - a| \leq 2\pi k\frac{\sqrt{a(1-a)}}{M} + k^2\frac{\pi^2}{M^2}, \tag{3.89}$$

con probabilità

$$\begin{cases} \geq \frac{8}{\pi^2} & \text{se } k = 1, \\ \geq 1 - \frac{1}{2(k-1)} & \text{altrimenti,} \end{cases}$$

dove M *è il numero di applicazioni della funzione black-box* f. *Se* $a = 0$, *allora* $\hat{a} = 0$, *certamente. Se* $a = 1$ *e* M *è pari, allora* $\hat{a} = 1$, *certamente.*

Dimostrazione. In QAE si applichi F_M, indi per cui lo stato risulta:

$$\frac{1}{\sqrt{2M}}\sum_{j=0}^{M-1}|j\rangle\left(e^{i\theta}|\psi_+\rangle - e^{-i\theta}|\psi_-\rangle\right). \tag{3.90}$$

Si applichi $\Lambda(\mathcal{Q})$:

$$\frac{1}{\sqrt{2M}} \sum_{j=0}^{M-1} |j\rangle \left(e^{i\theta} e^{2ij\theta} |\psi_+\rangle - e^{-i\theta} e^{-2ij\theta} |\psi_-\rangle \right)$$

$$= \frac{e^{i\theta}}{\sqrt{2M}} \sum_{j=0}^{M-1} e^{2ij\theta} |j\rangle |\psi_+\rangle - \frac{e^{-i\theta}}{\sqrt{2M}} \sum_{j=0}^{M-1} e^{-2ij\theta} |j\rangle |\psi_+\rangle$$

$$= \frac{e^{i\theta}}{\sqrt{2}} |S_M \left(\frac{\theta}{\pi}\right)\rangle |\psi_+\rangle - \frac{e^{-i\theta}}{\sqrt{2}} |S_M \left(1 - \frac{\theta}{\pi}\right)\rangle |\psi_-\rangle . \quad (3.91)$$

Si applichi F_M^{-1} al primo registro e poi si compia la misurazione del primo registro. Usando il Teorema (3.6.1), si evince che il risultato prodotto deriva dalla misurazione di $F_M^{-1} |S_M \left(\frac{\theta}{\pi}\right)\rangle$ oppure di $F_M^{-1} |S_M \left(1 - \frac{\theta}{\pi}\right)\rangle$. Per ciascuno si può dire che la probabilità di misurare $|y\rangle$ nel primo è uguale alla misurazione di $|M - y\rangle$ nel secondo. Da ciò segue direttamente che

$$\sin^2 \pi \frac{y}{M} = \sin^2 \pi \frac{(M - y)}{M}. \quad (3.92)$$

La stima dell'angolo θ risulta allora

$$\hat{\theta} = \pi \frac{y}{M} = \pi \frac{y}{2^m}. \quad (3.93)$$

Usando di nuovo il Teorema (3.6.1),

$$\mathbb{P} \left(d\left(\hat{\theta}, \theta\right) \leq \frac{k}{M} \right) \quad (3.94)$$

è inferiormente limitata per un certo valore in funzione di $k > 0$, ed M.
Siano $\hat{a} = \sin \hat{\theta}$ $a = \sin \hat{\theta}$. Usando Lemma (3.6.2) ed imponendo $\epsilon = \frac{\pi}{M}$, segue

$$|\hat{a} - a| \leq 2 \frac{\pi}{M} \sqrt{a(1 - a)} + \frac{\pi^2}{M^2}. \quad (3.95)$$

□

Da questo risultato si può osservare che lo speed-up quadratico, che era già stato introdotto da Grover per un problema di ricerca, viene mantenuto anche in questo contesto. Infatti, sapendo che il processo classico di stima di a consisterebbe in una simulazione Monte Carlo, allora l'errore che ne deriverebbe è dell'ordine di $\mathcal{O}\left(\frac{1}{\sqrt{M}}\right)$. M corrisponde alla cardinalità del campione. Invece, utilizzando QAE che sfrutta a sua volta una procedura di Amplitude Amplification, dall'osservazione di (3.95) si evince che l'errore di stima è $\mathcal{O}\left(\frac{1}{M}\right)$, ossia di un ordine quadraticamente inferiore.

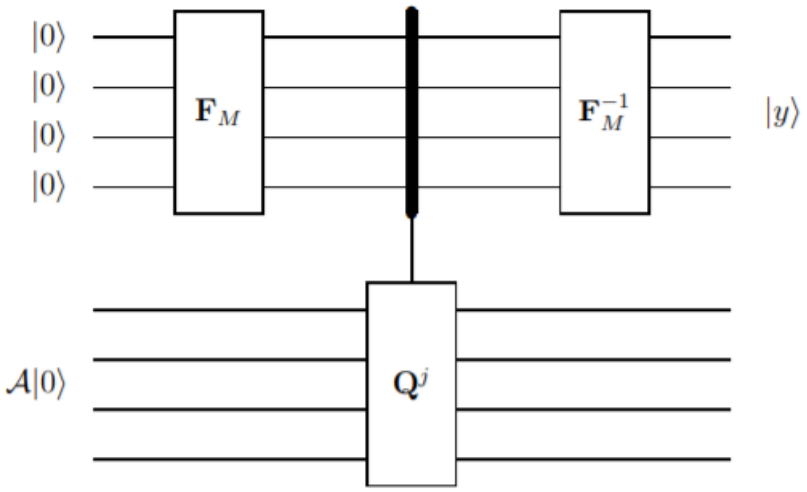


Figura 3.3: Circuito per Quantum Amplitude Estimation.

Gli steps di QAE (Figura 3.3) sono i seguenti:

1. inzializzare due registri di m e $n+1$ qubits allo stato totale $|0\rangle_m \mathcal{A}|0\rangle_{n+1}$;
2. applicare la trasformazione di Fourier F_M sul primo registro. Se M è una potenza di 2, esso corrisponde ad applicare una porta Hadamard ad ogni qubits;
3. applicare $\Lambda_M(\mathcal{Q})$ con $\mathcal{Q} = -\mathcal{A}S_0\mathcal{A}^{-1}S_{\psi_0}$;
4. applicare QFT inverso F_M^{-1} al primo registro;
5. eseguire un measurement al primo registro, che fornisce un risultato $|y\rangle$;
6. calcolare $\hat{a} = \sin^2\left(\frac{\pi y}{M}\right)$.

Capitolo 4

Credit Risk Analysis

4.1 Il rischio di credito

Gli accordi internazionali *Basel Accord* costituiscono un corpo di regolamentazioni stipulate da Basel Committee on Banking Supervision a partire dal 1988. Questa istituzione, con sede a Basilea, nacque per la supervisione dell'operato delle banche centrali mondiali, al fine di garantire un livello minimo di robustezza del settore bancario. La crisi finanziaria del 2008, che colpì in un primo momento il mercato statunitense e che poi sfociò in una recessione che interessò tutta l'economia mondiale, mise alla luce la vulnerabilità del sistema finanziario internazionale. Pertanto, in seguito a tale evento, venne accentuata maggiormente la politica di supervisione degli istituti di credito. Gli accordi decretarono l'inizio di una prassi di controllo delle posizioni contrattuali assunte dagli istituti, sia tramite una verifica interna, sia tramite processi di vigilanza esterni. Essi sono volti a garantire i requisti minimi di solvibilità e robustezza imposti dalle regolamentazioni, a fronte di ipotetici scenari di stress ed, in alcuni casi, addirittura catastrofici. Le minacce che indeboliscono ogni soggetto finanziario operante sul mercato sono di triplice natura: esso deve far fronte ad insperati rischi di mercato, creditizi e operativi.
Basilea III, attualmente in vigore, pone ampia attenzione sul rischio di credito. Gli istituti finanziari devono essere in grado di gestire le conseguenti perdite derivanti dal rischio di eventuali defaults dei loro contraenti. Infatti, il portafoglio di ciascun

ente finanziario è quotidianamente soggetto al rischio di default di controparti che hanno contratto un debito. Un approccio di cui usufruiscono per quantificare le misure di rischio viene identificato con la sigla *IRB* (*Internal-rated based*). Secondo questo metodo le banche posseggono la facoltà di stimare i propri parametri di controllo, che servono a calcolare il *capital requirement* e definire l'esposizione al rischio. Dopodiché tali stime vengono tradotte nei valori standard tramite delle operazioni costruite appositamente per ciascun istituto di credito. In questo modo la regolamentazione garantisce a ciascun ente di sfruttare metodi personalizzati di calcolo, ma limita la possibilità che tali stime risultino distorte eccessivamente, tramite l'intervento finale di un controllo superiore esterno.

4.2 Il modello ASRF

Asymptotic Single Risk Factor indica il modello matematico di rischio di credito implementato per l'approccio IRB. Il suo nome fa riferimento alle principali caratteristiche, per le quali, in questa sezione, vengono espressi i principi matematici di fondo. Quello che caratterizza maggiormente questo modello è la sua semplicità di calcolo. Inoltre, il rischio totale viene modellato in modo tale che esso dipenda solamente da ciascun asset e non dalla composizione del portafoglio.
Per capire come opera questo metodo, bisogna descrivere l'oggetto su cui vengono calcolate le misure di rischio. Nel contesto bancario, si può considerare un portafoglio di assets, ciascuno etichettato dal proprio valore corrente. La somma totale determina il valore del portafoglio che detiene l'istituto di credito. Gli assets corrispondono al valore degli investimenti in cui è frazionato il capitale iniziale della banca. In generale, si assume che il default di un soggetto finanziario dipenda direttamente dal valore del suo asset. In pratica esiste una soglia minima di valore, tipica di ciascuna azienda quotata operante sul mercato, tale che determina il limite di sopravvivenza alla fine di un determinato orizzonte temporale. Infatti, se il suo valore scende sotto tale soglia, allora si può asserire che l'azienda è in stato di default e quindi non è in grado di coprire il debito nei confronti della banca che lo ha erogato. Altrimenti, significa che il soggetto è ancora attivo ed il suo asset è presente sui listini delle borse in cui è quotato. Nel caso in cui un'azienda dichiari il

proprio fallimento, di norma si apre una procedura regolata da precise norme legali, che sono atte a spartire il valore residuo degli assets ricavato dalla liquidazione. Gli azionisti e gli investitori corrispondono all'anello finale della catena dei soggetti che hanno diritto ad una riscossione del debito.

Si indichi la probabilità di D_i, ossia l'evento default dell'i-esima azienda, i cui asset sono presenti nel portafoglio della banca, con $p_i = \mathbb{P}(D_i)$.
Il valore dell'asset della azienda i, indicato con A_i, segue una dinamica descritta tramite il modello di Merton, il cui autore ne ha presentato i principi in un articolo del 1974 [15]. Nel corso del tempo t, il valore dell'asset $A_i(t)$ segue un processo stocastico di moto geometrico Browniano:

$$dA_i(t) = \mu_i A_i(t)dt + \sigma_i A_i(t)dW_i(t), \tag{4.1}$$

dove μ_i è il *mean rate of return* dell'asset i, e σ_i è la sua volatilità. $W_i(t)$ è un moto Browniano, per cui vale la proprietà:

$$W_i(t) \sim \mathcal{N}(0, t). \tag{4.2}$$

Il processo, opportunamente integrato, si esprime con la seguente:

$$\log A_i(t) = \log A_i(0) + \mu_i t - \frac{1}{2}\sigma_i^2 t + \sigma_i \sqrt{t} W \qquad W_i = W_i(1) \sim \mathcal{N}(0,1), \tag{4.3}$$

da cui si ricava

$$A_i\,(t) = A_i\,(0)\, e^{\left(\mu_i - \frac{1}{2}\sigma_i^2\right)t + \sigma_i \sqrt{t} W_i}. \tag{4.4}$$

Il modello di Merton stima la probabilità di default di una singola azienda presa in considerazione separatamente dalle altre. Pertanto, d'ora in avanti si evita di specificare il pedice i delle variabili per non appesantire la notazione. Il modello di Merton si basa sulla formula di Black-Scholes-Merton [16]: infatti l'equity dell'azienda, di cui si vuole definire il default, viene modellato secondo il procedimento con cui si valuta un'opzione call europea.

Definizione 4.2.1. *L'equity di una soggetto economico è definito come*

$$E = A - L, \tag{4.5}$$

cioè la differenza tra il valore di tutte le proprietà detenute dall'azienda A e i debiti, o liabilities, che le stanno a carico L.

Il parallelo con le opzioni call è semplice: le liabilities assumono il significato che nel BSM ricopre lo strike price, mentre gli asset corrispondono al valore corrente del sottostante dell'opzione. Dunque, se il valore degli assets, entro un certo orizzonte temporale T, che corrisponde alla maturità dell'opzione, è inferiore a quello del debito totale contratto dall'azienda, allora significa che essa non è in grado di onorare il pagamento del proprio debito.
Si consideri il valore dell'asset A, che segue il processo stocastico definito dalla (4.4). Tuttavia in questo contesto viene calcolato rispetto alla misura di probabilità neutrale al rischio $\mathcal{Q}$. Quindi, si ottiene:

$$A\left(t\right) = A\left(0\right) e^{\left(r-\frac{1}{2}\sigma^2\right)t+\sigma\sqrt{t}W}, \tag{4.6}$$

dove r è il tasso risk-free. Ciò permette di scontare il valore atteso futuro dell'opzione in modo tale che esso coincida con il valore dell'asset al tempo presente. In altre parole, il valore dell'asset deve rispettare la proprietà di martingala e ciò si ottiene facendo coincidere il drift del derivato con il tasso risk-free. Sotto $\mathcal{Q}$ non sono permessi arbitraggi, ed è questa la motivazione per cui questo procedimento viene utilizzato in un contesto di pricing. Pertanto, il modello di Merton stima il valore dell'equity dell'azienda al tempo T come

$$E\left(T\right) = \max\{A\left(T\right) - L, 0\}. \tag{4.7}$$

Il BSM model restituisce come valore dell'equity scontato al tempo iniziale

$$E_0 = A_0\Phi(d_1) - Le^{-rT}\Phi\left(d_2\right) \tag{4.8}$$

dove

$$E_0 = E\left(0\right) \qquad A_0 = A\left(0\right), \tag{4.9}$$

$$d_1 = \frac{\ln A_0/L + \left(r + \sigma_A^2/2\right)T}{\sigma_A\sqrt{T}} \qquad d_2 = d_1 - \sigma_A\sqrt{T}, \tag{4.10}$$

σ_A è la volatilità dell'asset e Φ è la funzione cumulativa di probabilità della normale standard.

Il valore del debito al tempo iniziale $t = 0$ è dato da $A_0 - E_0$.
La probabilità che l'opzione call sia *in the money*, ossia in grado di generare un payoff positivo alla maturità tramite la sua esercitazione, è data da $\Phi(d_2)$. Quindi, la probabilità che l'azienda sia in stato di default al tempo T è uguale a:

$$p = \Phi(-d_2). \tag{4.11}$$

Per calcolarlo, occorre conoscere A_0 e σ_A, i quali non sono facilmente ricavabili. Invece, è possibile conoscere il valore dell'equity inziale E_0, se l'azienda è quotata in borsa perchè questo dato è reso pubblico. Quindi, una condizione che deve essere rispettata in funzione di A_0 e σ_A, è data dall'equazione (4.8). Per determinare numericamente entrambi i valori, è necessaria una seconda condizione, che viene fornita grazie al lemma di Ito:

$$\sigma_E E_0 = \frac{\delta E}{\delta A}\sigma_A A_0 = \Phi(d_1)\,\sigma_A A_0. \tag{4.12}$$

Una volta risolto il sistema delle due equazioni (4.8), (4.12) nelle due incognite A_0 e σ_A, si può determinare la probabilità di default alla maturità T, ossia $p = \Phi(-d_2)$.

Il modello di Merton fornisce un metodo analitico per stimare la probabilità incondizionata di default di una singola azienda. Queste stime sono fornite dalle agenzie di rating e vengono utilizzate come input per modelli più sofisticati rispetto a quello di Merton. Questi ultimi ne costituiscono le estensioni, in cui si compie la generalizzazione al caso in cui il numero di aziende del portafoglio è maggiore di uno.
Una di queste estensioni coincide proprio con l'ASRF model, sviluppato da Vasicek nel 2002. Egli ebbe l'idea di descrivere la distribuzione delle perdite su un portafoglio di azioni in modo tale da dividere il rischio totale del portafoglio su due tipi di fattori: uno di essi è comune a tutti gli assets e fa riferimento all'andamento che segue il mercato generale, mentre l'altro è tipico delle fluttuazioni di ogni singolo asset.
Innanzitutto, si può riassumere in un altro modo l'evento di default del singolo asset:

$$D_i = \{W_i < \Phi^{-1}(p_i)\}, \tag{4.13}$$

dove la variabile latente normale standard W_i viene utilizzata per modellare l'evento di default dell'asset i. Mentre precedentemente si è utilizzato L per indicare il debito totale contratto da una azienda, ora si indichi con L_n la successione di variabili casuali che indicano le perdite percentuali, entro un certo orizzonte temporale T, degli n assets che formano il portafoglio. Dopodiché, si introducono due parametri di interesse, che rientrano nell'insieme di valori su cui si basano le analisi dall'approccio IRB:

- *exposure at default*: EAD è una misura di rischio di esposizione di un certo credito rispetto ad un preciso debitore. Essa viene indicata con $\delta_i \in \mathbb{R}_+$ e si riferisce all'i-esimo contraente del credito;
- *loss given default*: LGD è il valore atteso in percentuale del credito che non è possibile ottenere in caso di default dalla liquidazione dell'asset, sia in via giudiziale sia stragiudiziale. $\eta_i \in [0, 1]$.

La perdita percentuale totale del portafoglio è

$$L_n = \sum_{i=1}^{n} \omega_i \eta_i \mathbb{1}_{\{D_i\}} \tag{4.14}$$

dove:

- $w_i = \frac{\delta_i}{\sum_{j=1}^{n} \delta_i}$ è l'esposizione del debitore i pesata su quella totale;
- $\mathbb{1}_{D_i}$ è l'indicatore dell'evento di default.

La variabile latente W_i, che modella la variabilità del valore dell'asset i, viene rappresentata come una combinazione lineare di altre variabili casuali e definita in questo modo:

$$W_i = \sqrt{\rho_i} Z + \sqrt{1 - \rho_i} Y_i, \tag{4.15}$$

dove Y_i per $i = 1, \ldots, n$ e Z sono i.i.d. $\mathcal{N}(0,1)$. Quello che segue direttamente da tale assunzione è che gli assets i e j possiedono correlazione $E[R_i, R_j] = \sqrt{\rho_i \rho_j}$ e la correlazione che esiste tra l'asset i e il fattore sistematico Z è data da $\sqrt{\rho_i}$. Pertanto, l'interpretazione che si da a $\rho_i \in (0,1), i = 1, \ldots, n$ è di un indice di sensibilità dell'asset i rispetto a Z ed esso viene determinato tramite le serie storiche di W_i,

Y_i e Z. In questo modo, la variabile latente W_i che indica la variabilità del valore dell'asset, viene ulteriormente modellata tramite un *modello ai fattori* nelle due seguenti entità:

- *rischio sistematico*, Z: è il rischio che tiene conto del cambiamento dell'intero mercato. Esso è comune a tutti gli elementi del portafoglio e costituisce il rischio non eliminabile dell'intera distribuzione di guadagno di un portafoglio;
- *rischio idiosincratico*, Y_i: è tipica di ogni asset, infatti viene anche definito *rischio specifico*. In teoria di ottimizzazione di portafoglio si è dimostrato che l'effetto di tale rischio può essere annullato tramite una o più opportune configurazioni di portafoglio.

Dunque, la variabile di default del soggetto debitore i assume la distribuzione di una variabile bernoulliana. Cioè, essa può assumere valore uguale a 1 con probabilità p_i e 0 con probabilità $1 - p_i$. La stima di probabilità incondizionata di default p_i viene modellata tramite il seguente procedimento. Si sostituisca W_i, definita come in (4.15), in (4.13). Allora, la probabilità di default dell'asset i, condizionata sul rischio sistematico Z, diventa

$$\begin{aligned} p_i(z) = \mathbb{P}(D_i|Z=z) &= \mathbb{P}\left(W_i < \Phi^{-1}(p_i)\,|Z=z\right) \\ &= \mathbb{P}\left(\sqrt{\rho_i}z + \sqrt{1-\rho_i}Y_i < \Phi^{-1}(p_i)\right) \\ &= \mathbb{P}\left(Y_i < \frac{\Phi^{-1}(p_i) - \sqrt{\rho_i}z}{\sqrt{1-\rho_i}}\right) \\ &= \Phi\left(\frac{\Phi^{-1}(p_i) - \sqrt{\rho_i}z}{\sqrt{1-\rho_i}}\right) \end{aligned} \tag{4.16}$$

dove p_i è la probabilità non condizionata di default dell'asset i. Si noti che il valore atteso della perdita derivante dall'asset i condizionato a Z, in accordo con le proprietà della variabile bernoulliana con cui si modella, è uguale alla probabilità di default del medesimo:

$$\mathbb{E}\left[L_i|\, Z=z\right] = p_i(z) = \Phi\left(\frac{\Phi^{-1}(p_i) - \sqrt{\rho_i}z}{\sqrt{1-\rho_i}}\right) \tag{4.17}$$

Applicando la funzione inversa della CDF per una normale standard, si ottiene il quantile dipendente dal rischio sistematico, $\zeta_i(z)$, che determina il limite di

probabilità di default. Ossia,

$$\zeta_i(z) = \frac{\Phi^{-1}(p_i) - \sqrt{\rho_i}z}{\sqrt{1-\rho_i}} = \Phi^{-1}(p_i(z)) \quad i = 1, \ldots, n \tag{4.18}$$

per una certa realizzazione z di Z. Quindi, data $Z = z$, la perdita percentuale totale è

$$L_n = \sum_{i=1}^{n} \omega_i \eta_i \mathbb{1}_{\{Y_i < \zeta_i(z)\}}. \tag{4.19}$$

L'estensione ASRF tratta le probabilità condizionate $p_i(z)$ su un singolo rischio sistematico Z, distinguendosi del modello di Merton, il quale considera probabilità incondizionate p_i.
La probabilità della perdita di portafoglio è data dal valore atteso delle perdite dato un certo valore z del fattore sistematico. A livello pratico, ciò si traduce assumendo alcuni scenari di realizzazione di Z, per ciascuno dei quali si calcola la probabilità di un certo valore di perdita ed, infine, si pesa ogni scenario per la sua probabilità.
Il valore atteso delle perdite totali calcolato su (4.14) è

$$\mathbb{E}[L_n] = \sum_{i=1}^{n} \omega_i \eta_i p_i, \tag{4.20}$$

mentre il valore atteso condizionato calcolato su (4.19) è

$$\mathbb{E}[L_n | Z = z] = \sum_{i=1}^{n} \omega_i \eta_i p_i(z). \tag{4.21}$$

Per illustrare la definizione del carattere asintotico del modello ASRF, si definisca $\Delta = \sum_{k=1}^{\infty} \delta_k$, ossia una successione infinita di EAD, relativa ai debitori di un portafoglio. Allora la somma parziale di Δ di ordine n è definita come $\Delta_n = \sum_{k=1}^{n} \delta_k$.

Definizione 4.2.2. *Un portafoglio è detto asintotico se rispetta tale proprietà:*

$$\sum_{n=1}^{\infty} \left(\frac{\delta_n}{\Delta_n}\right)^2 < \infty. \tag{4.22}$$

Questa condizione permette al portafoglio di raggiungere un determinato livello di granularità tale che l'esposizione al rischio per un singolo asset influenzi in modo trascurabile la distribuzione delle perdite totali. In altre parole, la distribuzione delle perdite non è influenzata da un asset in particolare. Inoltre, questa assunzione permette al modello di essere perfettamente additivo: il capitale richiesto per far fronte ad un debito dipende solo dalla rischiosità di tale credito e non da come è strutturato il portafoglio.

4.3 Le misure di rischio: VaR e CVaR

Definito il modello di rischio con cui si stima la perdita totale del portafoglio, è opportuno trattare le misure di rischio che vengono calcolate su tale oggetto.

Definizione 4.3.1. *Una misura di rischio è un funzionale*

$$g : X(\omega) \to \mathbb{R} \qquad \forall \omega \in \Omega, \tag{4.23}$$

cioè un operatore che mappa una variabile casuale in $\mathbb{R}$ *o in un suo sottoinsieme.*

In ambito di Risk Management, è importante classificare tali misure a seconda che esse soddisfino la *proprietà di coerenza*. Tale caratteristica porta alcuni vantaggi: ad esempio, una misura di rischio coerente può essere utilizzata in problemi di ottimizzazione. In altre parole, se una misura è coerente allora essa rispetta la nozione di convessità e quindi, assunta come funzione obbiettivo di un problema di ottimo, restituisce una soluzione globale.

Definizione 4.3.2. *Una misura di rischio è coerente se rispetta le seguenti proprietà:*

- ***monotonia**: al crescere della realizzazione della variabile casuale cresce (oppure diminuisce) anche la misura di rischio associata. Se la variabile tiene traccia delle perdite allora la misura di rischio è crescente. Se invece la variabile è relativa ai guadagni, allora è decrescente. Una conseguenza importante è che, per due differenti realizzazioni della stessa variabile casuale, la misura non fornisce lo stesso output;*

- ***omogeneità positiva****: sia x una realizzazione di X allora*

$$g(bx) = bg(x) \qquad \forall b \in \mathbb{R}; \tag{4.24}$$

- ***invarianza per traslazioni****: sommando una quantità fissa alla variabile casuale, la misura di rischio aumenta della medesima quantità, ossia*

$$g(x + c) = g(x) + c \qquad \forall c \in \mathbb{R}; \tag{4.25}$$

- ***subadditività****: combinando due variabili casuali, il rischio totale diminuisce. Date X e Y fattori di rischio,*

$$g\left(X + Y\right) \leq g(X) + g(Y). \tag{4.26}$$

La principale misura di rischio, secondo cui gli Accordi di Basilea basano le procedure di calcolo dei requisiti di capitale minimo, è il *Value at Risk*.

Definizione 4.3.3. *Detto anche VaR, oppure anche V@R, il Value at Risk è un quantile della distribuzione del fattore di rischio X. Assumendo un certo livello di confidenza $1 - \alpha \in [0,1]$, allora*

$$VaR_{1-\alpha} = \inf\{x \in \mathbb{R} : \mathcal{P}\left(X > x\right) \leq \alpha\}. \tag{4.27}$$

Il VaR non è coerente, in quanto rispetta tutte le proprietà richieste a meno della subadditività. Quindi, può capitare che la somma del VaR calcolato su due distinti fattori di rischio X e Y sia minore del VaR calcolato sulla loro somma $X + Y$, cioè non viene rispettata la condizione:

$$VaR_{1-\alpha}(X + Y) \leq VaR_{1-\alpha}(X) + VaR_{1-\alpha}(Y). \tag{4.28}$$

Dunque, il VaR è una misura di rischio che, nel caso di un portafoglio azionario, non sempre premia la diversificazione degli investimenti.

Ritornando alla considerazione del modello ASRF, il VaR viene calcolato sulla distribuzione delle perdite totali:

$$VaR_{1-\alpha} = \inf\{x \in \mathbb{R} : \mathcal{P}\left(L_n > l\right) \leq \alpha\}. \tag{4.29}$$

Esso richiede la specificazione iniziale di due parametri:

- l'orizzonte temporale su cui viene calcolata la variazione del valore degli assets. Nel caso di Basilea si intende un intervallo di 10 giorni;

- la definizione di un livello di confidenza, che secondo gli accordi equivale a 99.9%.

Il VaR, per quanto goda di un ruolo centrale nelle analisi di Risk Management, soffre di alcune carenze operative, che per alcuni obbiettivi risulta essere limitato:

- non è coerente, quindi non è una misura convessa e non può essere utilizzata per la risoluzione di problemi convessi;

- non fornisce alcuna informazione sulle reali conseguenze dovuta ad una realizzazione della variabile casuale campionata nella regione di rischio. Di solito, questo evento appartiene ad una coda dela distribuzione.

Per cercare di risolvere questi due aspetti, spesso è necessario considerare a supporto del VaR una seconda misura di rischio.

Definizione 4.3.4. *Il CVaR, detto anche Expected Shortfall (ES), è una misura di rischio coerente che deriva dal VaR e, definito un livello di confidenza* $1-\alpha$ *e un orizzonte temporale* T*, si calcola nel seguente modo:*

$$CVaR_{1-\alpha}(X) = \mathbb{E}[X|X > VaR_{1-\alpha}(X)]. \tag{4.30}$$

L'expected shortfall è una misura di rischio coerente che deriva dal VaR calcolato allo stesso livello di confidenza $1-\alpha$. Esso corrisponde al valore atteso del fattore di rischio calcolato sui valori estremi della coda individuata dal VaR. Infatti, per definizione esso è un valore atteso condizionato.
Definita f_x la funzione di densità di probabilità di X, valgono le seguenti equazioni:

$$ES_{1-\alpha}(X) = \mathbb{E}[X|X > VaR_{1-\alpha}(X)] = \frac{\sum_{i=VaR_{1-\alpha(X)}+1}^{\infty} x f_x}{\alpha}, \tag{4.31}$$

oppure

$$ES_{1-\alpha}(X) = \mathbb{E}[X|X > VaR_{1-\alpha}(X)] = \frac{\int_{VaR_{1-\alpha(X)}}^{\infty} x f_x}{\alpha}, \tag{4.32}$$

se la variabile di rischio segue una distribuzione discreta o continua, rispettivamente. Di norma, il procedimento seguito per le stime di queste misure di rischio viene effettuato tramite una simulazione Monte Carlo. Inizialmente, è opportuno caratterizzare le distribuzioni dei fattori di rischio tramite un modello; dopodiché, attraverso la generazione di numeri pseudocasuali, si ottengono gli scenari di calcolo. Ovviamente la grandezza del campione di simulazione determina la precisione dei risultati: l'ampiezza degli intervalli di confidenza sulle stime delle statistiche di interesse scala come $\frac{1}{\sqrt{N}}$, dove N è la cardinalità del campione. Inoltre, esistono delle tecniche di riduzione della varianza, che permettono di ottenere delle stime più precise di tali statistiche, senza ricorrere necessariamente ad un numero eccessivamente elevato del campione.
Infine, occorre tenere presente che, oltre alla varianza tipica della simulazione, anche il modello costituisce un fattore di rischio.

In questo capitolo è stato presentato il modello ASRF con cui si stima la distribuzione empirica delle perdite associato ad un portafoglio azionario. Inoltre, sono state introdotte le misure di rischio VaR ed Expected Shortfall, che rivestono un ruolo centrale per il calcolo del capitale di liquidità che gli istituti di credito devono essere in grado di garantire. Nel prossimo capitolo viene affrontato l'algoritmo del caso di analisi, presente in Qiskit. Si ricorda che lo scopo di questa tesi, infatti, è mostrare come le analisi del rischio di credito possano essere effettuate tramite l'utilizzo di nuovi strumenti di calcolo. Il prossimo capitolo verte sulla codifica su hardware quantistico del modello ASRF e del calcolo delle statistiche di interesse, come il valore atteso ed il VaR.

Capitolo 5

Il caso di applicazione

Questo capitolo rappresenta il cuore della tesi. Finora è stata fatta un'ampia introduzione al mondo computazionale del Quantum Computing, in cui sono stati presentati gli attori principali e le leggi su cui verte il suo funzionamento. Dopodichè, è stata affrontata la teoria che riguarda il caso di applicazione: la Credit Risk Analysis, all'interno di un contesto finanziario.
Successivamente, seguendo il lavoro presentato da Stefan Worner e Daniel J. Egger in [2], è possibile capire il modo in cui il Quantum Computing può essere applicato ad un caso specifico per il calcolo di determinate misure di rischio. Infatti, IBM concede libero accesso alla propria piattaforma Qiskit [1] e la possibilità di poter conoscere i *cases* sviluppati dai suoi ricercatori negli svariati campi di applicazione del QC. Oltre al caso di Credit Risk Analysis, esiste una serie di altri problemi che appartengono al mondo finanziario: per citare un esempio, è presente un Tutorial in cui si implementa un meccanismo quantistico di pricing per un derivato standard, come lo è un'opzione call europea.
Ovviamente questi cases sono di fresca implementazione e, come spesso IBM si preoccupa di specificare, non intendono sostituire nel breve termine i metodi sofisticati che la teoria classica ha sviluppato per la risoluzione dei problemi reali. Infatti, lo stato dell'arte dell'hardware quantistico non è attualmente in grado di sostenere lo sforzo necessario per produrre una soluzione efficace e soprattutto precisa. Però,

allo stesso tempo, è possibile sviluppare, almeno teoricamente, gli algoritmi quantistici che, insieme ai miglioramenti futuri delle tecnologie quantistiche, potranno costituire un valido metodo di risoluzione.
Pertanto, questo capitolo affronta il case Credit Risk Analysis, per il quale è necessario conoscere le nozioni delle sezioni precedenti. L'obbiettivo è dimostrare che, tramite l'utilizzo dell'algoritmo Quantum Amplitude Estimation, è possibile ottenere uno speedup quadratico rispetto alle tecniche classiche di campionamento Monte Carlo. Nella fattispecie, QAE è sfruttato per calcolare il valore atteso ed il VaR sulla distribuzione di una variabile casuale che costituisce il fattore di rischio di monitoraggio. In ambiente classico, l'errore circa le loro stime scala come $\mathcal{O}(\frac{1}{N})$, dove N è la numerosità del campione generato dalla distribuzione teorica. Invece, come è stato dimostrato nel Capitolo 4, le stime calcolate con Monte Carlo possiedono un'imprecisione $\mathcal{O}(\frac{1}{\sqrt{N}})$.
Nel seguito, sono riportate le sezioni in cui è diviso l'algoritmo. Invece, in Appendice $A.1$ è reperibile il codice scritto tramite istruzioni Python che fanno uso di oggetti e funzioni appartenenti alla libreria Qiskit.

5.1 Il modello

Come lo è per un generico algoritmo quantistico, è di fondamentale importanza il ruolo che svolge il vettore di stato del sistema. Esso è l'oggetto su cui vengono caricate tutte le informazioni che è utile memorizzare per la codifica del modello su hardware quantistico. Partendo da uno stato iniziale, settato ad un valore prefissato, tramite una serie di trasformazioni lineari, si giunge al processo di misurazione, che fornisce la stima dei risultati di interesse.
Quindi, è opportuno specificare quali sono gli oggetti che sono trattati nel case e quale modello urge implementare su QC. Sia dato un portafoglio di K assets, le cui probabilità di default $p_k(z)$ seguono un modello ASRF, ampiamente descritto nel capitolo precedente. Gli eventi di default sono condizionatamente indipendenti rispetto ad un fissato valore della variabile sistematica Z, che funge da indice di mercato. Si assume che quest'ultimo segua una distribuzione normale standard,

$Z \sim \mathcal{N}(0,1)$.
Detta Φ la CDF di Z, allora

$$p_k(z) = \Phi\left(\frac{\Phi^{-1}(p_i) - \sqrt{\rho_i}z}{\sqrt{1-\rho_i}}\right). \tag{5.1}$$

Quindi la distribuzione di perdita totale è data da

$$L = \sum_k \lambda_k X_k, \tag{5.2}$$

dove X_k è una variabile bernoulliana con probabilità di successo uguale alla (5.1). In questo caso, contrariamente a quanto definito dalla (4.19), il parametro definito da λ_k tiene conto del prodotto di LGD, definita nel paragrafo precedente di natura percentuale, e di EAD, di natura assoluta. In questo modo, esso rappresenta il valore atteso della quantità assoluta di capitale non più recuperabile in caso di default dell'asset k.
Innanzitutto, è opportuno definire quali sono i parametri che bisogna specificare all'inizio dell'implementazione:

- n_z, il numero di qubits utilizzato per implementare Z;
- z_{max}, il numero intero massimo con il quale si discretizza il dominio di campionamento di Z;
- p_0^k, la probabilità di default dell'asset k quando $z = 0$;
- ρ_k, la sensibilità di X_k rispetto a Z;
- λ_k, ossia la loss given default dell'asset k;
- α, che determina il livello di confidenza prescelto $1 - \alpha$.

Per chiarire il senso delle prime due variabili, si definisca un registro quantistico *Z-register* composto da n_z bits. In questo modo gli stati totali che esso può rappresentare sono in totale 2^{n_z}. Questi sono tutti i possibili valori che può assumere la variabile latente Z. Si compie così una discretizzazione della sua distribuzione, che per definizione è continua: si ricorda infatti che $Z \sim \mathcal{N}(0,1)$.
z_{max} corrisponde al massimo valore di discretizzazione che può assumere Z in

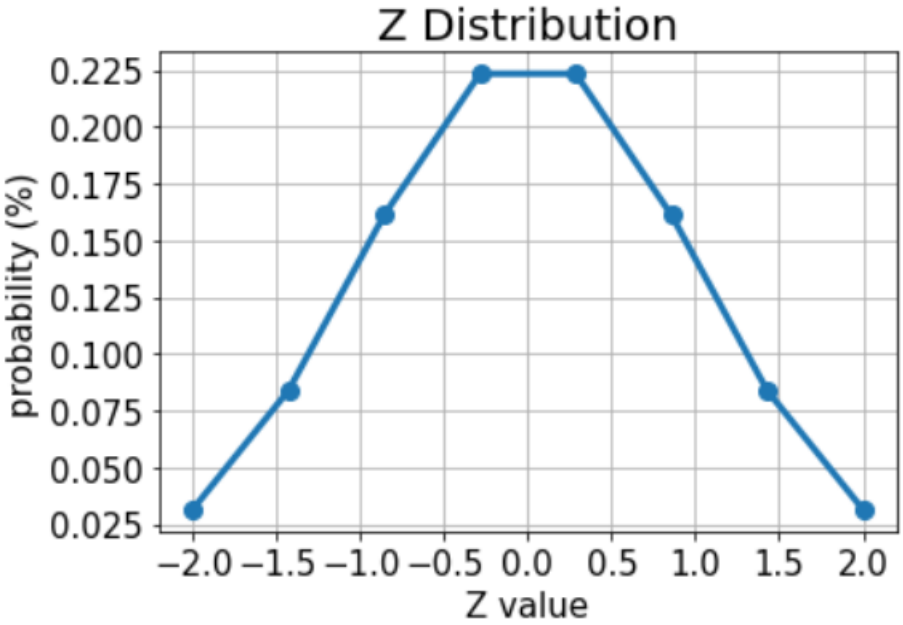


Figura 5.1: La PDF viene discretizzata in un insieme di 2^{n_z} punti a partire da $-z_{max}$ fino a $+z_{max}$. In questo esempio, $n_z = 3$ e $z_{max} = 2$. Per la distribuzione normale standard, con PDF ϕ, vale $\phi(2) = 0.9772$.

modulo sulle proprie code estreme. In altri termini, Z può assumere 2^{n_z} valori equidistanti all'interno dell'intervallo $[-z_{max}, z_{max}]$ (Figura 5.1). Ovviamente, siccome la variabile è gaussiana, si preferisce centrare tale intervallo sui valori massimi della PDF dove è presente la densità maggiore. In realtà, portandosi al caso più generale, si crea una mappa affine

$$z_i = a_z i + b_z \tag{5.3}$$

nei coefficienti a_z, b_z che dipendono dalla discretizzazione, con $i \in \{0, \ldots, 2^{n_z} - 1\}$. Nel caso descritto precedentemente valgono $b_z = -z_{max}$ e $a_z = \frac{z_{max}}{2^{n_z-2}}$.
Pertanto, la distribuzione di probabilità è codificata sul vettore di stato del sistema tramite un apposito operatore $\mathcal{R}$, che permette di conferire ad ogni z_i la propria amplitudes $\sqrt{p_i}$, secondo la PDF di Z. Dunque, dato uno stato iniziale $|0\rangle$ per tutti i qubits, si esegue:

$$\mathcal{R} |0\rangle_{n_z} = |\psi\rangle_{n_z} = \sum_{i=0}^{N-1} \sqrt{p_i} |i\rangle_{n_z} , \tag{5.4}$$

con $\sum_{i=0}^{N-1} p_i = 1$, $p_i \in [0,1] \quad \forall i$ e dove $|\rangle_{n_z}$ indica un vettore n_z dimensionale. Chiamiamo un secondo registro *X-register* di K qubits, tanti quanti sono gli assets: gli eventi $\{X_1, \ldots, X_K\}$ sono non correlati per ogni campionamento z di Z [17]. Indi per cui è possibile codificare X_k del k-esimo asset sullo stato del qubit di corrispondenza facendo uso di una Y-rotation $R_Y\left(\theta_p^k\right)$ con angolo

$$\theta_p^k = 2 \arcsin\left(\sqrt{p_k}\right). \tag{5.5}$$

Il gate Y-rotation permette di compiere una rotazione di un angolo specificato attorno all'asse Y della sfera di Bloch. In generale, dato un angolo β, vale la seguente:

$$R_Y(\beta) = e^{-i\frac{\beta}{2}Y} = \cos\left(\frac{\beta}{2}\right) I - \sin\left(\frac{\beta}{2}\right) Y = \begin{pmatrix} \cos\left(\frac{\beta}{2}\right) & -\sin\left(\frac{\beta}{2}\right) \\ \sin\left(\frac{\beta}{2}\right) & \cos\left(\frac{\beta}{2}\right) \end{pmatrix}$$

Così facendo, l'operatore che carica sul registro X-register la distribuzione di default per ogni asset è

$$\mathcal{U} = \bigotimes_{k=1}^{K} R_Y\left(\theta_p^k\right), \tag{5.6}$$

Ciò prepara il qubit k allo stato bidimensionale

$$\sqrt{1-p_k}\,|0\rangle + \sqrt{p_k}\,|1\rangle\,, \tag{5.7}$$

la cui distribuzione di misurazione corrisponde esattamente a quella della variabile bernoulliana del modello. Ossia, la probabilità che ci sia un default per l'asset k è data tramite la misurazione dello stato $|1\rangle$ con probabilità della relativa amplitude elevata al quadrato, p_k.

A questo punto, considerando i due registri definiti finora, in cui si contano la somma totale di $n_z + K$ qubits, è opportuno definire il vettore di stato che descrive i movimenti del sistema meccanico. Esso è sempre inizializzato allo stato $|0\rangle$ per ogni qubit, come esige lo standard di programmazione Qiskit. Dopo l'esecuzione degli operatori $\mathcal{R}$ e $\mathcal{U}$ sui rispettivi registri di azione, il vettore di stato è il seguente:

$$|\psi\rangle = \sum_{i=0}^{N-1} \sqrt{p_i}\,|i\rangle_n \bigotimes_{k=1}^{K} \left(\sqrt{1-p_k(z_i)}\,|0\rangle + \sqrt{p_k(z_i)}\,|1\rangle\right), \tag{5.8}$$

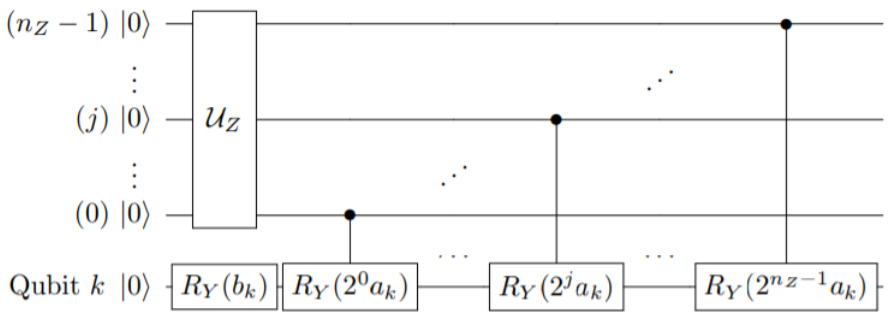


Figura 5.2: Circuito di rotazione del k esimo qubit appartenente al X-register. La rotazione è ottenuta tramite una serie di consecutive subrotazioni additive di minor fase, le quali sono controllate dal valore di ciascun qubit del Z-register.

dove il valore assunto da Z controlla la Y-rotation del singolo qubit k del X-register. Questo controllo avviene tramite il circuito rappresentato in Figura 5.2. L'angolo definito dalla (5.5) dipende dal valore assunto dalla variabile Z

$$\theta_p^k(z_i) = 2 \arcsin\left(\sqrt{p_k(z_i)}\right). \tag{5.9}$$

approssimato secondo uno sviluppo di Taylor al primo ordine secondo la mappa affine (5.3). Ne consegue la seguente:

$$\theta_p^k(z_i) \simeq a_z i + b_z \tag{5.10}$$

Fin qui, il modello risulta caricato sui registri iniziali: il vettore di stato $|\psi\rangle$ è stato fornito di tutta l'informazione necessaria per poter descrivere la dinamica delle X_k variabili bernoulliane, in dipendenza dai valori assunti dalla variabile latente. Ora, è necessario costruire gli operatori che contribuiscono al calcolo delle statistiche di interesse circa la distribuzione di total loss L.
Per verificare che la struttura dei circuiti sia corretta, grazie ad una prima fase in cui si sfrutta lo *Statevector Simulator*, è possibile simulare deterministicamente tutte le realizzazioni del vettore di stato che appartengono al dominio di misurazione. Dunque, sapendo che il numero di qubits totali è dato dalla somma dei due registri, $n_z + K$, segue che il vettore di stato $|\psi\rangle$ è $(n_z + K)$-dimensionale ed esistono $2^{(n_z+K)}$ possibili realizzazioni. Per ognuna di esse si conosce la perdita conseguente sul portafoglio totale e la relativa probabilità di misurazione. Tutto

ciò serve per determinare la distribuzione delle perdite totali. Su di essa si può calcolare analiticamente il valore esatto delle statistiche, che, successivamente, verrà impiegato per verificare la correttezza del circuito quantistico.

5.2 La stima del valore atteso

Per calcolare il valore atteso della perdita totale, non necessariamente viene fatto ricorso all'esecuzione di un circuito quantistico. Allo stesso modo, in un contesto computazionale classico, non viene per forza eseguita una simulazione Monte Carlo per calcolare $\mathbb{E}$[L]. Infatti, quando questa stima è facilmente ottenibile tramite una procedura analitica, allora risulta più efficiente proseguire per questa strada. In questo caso, secondo le proprietà del valore atteso, vale la seguente:

$$\mathbb{E}[L] = \mathbb{E}\left[\sum_{k=1}^{K} \lambda_k X_k\right] = \sum_{k=1}^{K} \lambda_k p_k. \tag{5.11}$$

Tuttavia, per il prosieguo del case in questione, è fondamentale costruire un operatore quantistico che carichi su circuito quantistico la perdita totale. Dunque, si definisca un nuovo registro *sum-register*, costituito da un numero di qubits uguale a

$$n_s = \lfloor \log_2 \lambda_1 + \cdots + \lambda_K \rfloor + 1. \tag{5.12}$$

Dopodiché, su di esso agisca un operatore $\mathcal{S}$, definito *weighted sum operator*:

$$\mathcal{S} : |x_1, \ldots, x_K\rangle_K |0\rangle_{n_s} \to |x_1, \ldots, x_K\rangle_K |\lambda_1 x_1 + \cdots + \lambda_K x_K\rangle_{n_s}, \tag{5.13}$$

dove $x_k \in \{0,1\}$ è la realizzazione di X_k.
$\mathcal{S}$ si occupa di trasportare dal K-register sul *sum*-register la somma totale delle perdite. La (5.12) è la cardinalità di quest'ultimo, che garantisce di poter caricare sul circuito il numero massimo di perdita totale [18]. Per l'implementazione di questo operatore si faccia riferimento ad Appendice B. La cardinalità massima di tale codifica è data da

$$S = 2^{n_s}, \tag{5.14}$$

in quanto il substato ristretto al solo *sum*-register $|\eta\rangle_{n_s}$ è il risultato di una mappatura da tutti i valori possibili di L in

$$|\eta\rangle_{n_s} = \sum_{i=0}^{S-1} \sqrt{p_i}\,|i\rangle_{n_s}\,, \tag{5.15}$$

con $|i\rangle \in \{0,1\}^{n_s}$ e p_i è la rispettiva probabilità di misurazione.
Dopodiché, una volta caricata la distribuzione delle perdite totali sul rispettivo registro, si definisce una funzione

$$f : \{0,\dots,S-1\} \to [0,1] \tag{5.16}$$

alla quale è associata un operatore

$$F : |i\rangle_{n_s}|0\rangle \to |i\rangle\left(\sqrt{1-f(i)}\,|0\rangle + \sqrt{f(i)}\,|1\rangle\right) \tag{5.17}$$

per $i \in \{0,\dots,S-1\}$. Come si può vedere dalla sua forma analitica, il valore dei qubits del *sum*-register controllano la realizzazione su un singolo ancilla qubit. Per quest'ultimo risulta che la probabilità di misurazione nello stato $|0\rangle$ è uguale a $1-f(i)$, mentre nello stato $|1\rangle$ equivale a $f(i)$.
Applicando F al sottostato (n_s+1)-dimensionale, che include il *sum*-register e l'ancilla qubit, si ottiene il seguente sottostato:

$$F(|\eta\rangle_{n_s}|0\rangle) = \sum_{i=0}^{S-1}\left(\sqrt{1-f(i)}\sqrt{p_i}\,|i\rangle_{n_s}|0\rangle + \sqrt{f(i)}\sqrt{p_i}\,|i\rangle_{n_s}|1\rangle\right). \tag{5.18}$$

Si noti che la probabilità di misurare $|1\rangle$ nel ancilla qubit è uguale a

$$\sum_{i=0}^{S-1}\sqrt{f(i)}\sqrt{p_i}, \tag{5.19}$$

che corrisponde proprio a

$$\sum_{i=0}^{S-1} f(i)p_i = \mathbb{E}\left[f(L)\right]. \tag{5.20}$$

Quindi, se si sceglie di implementare un operatore F, tale che la funzione associata f data dalla (5.16) risulti

$$f(i) = \frac{i}{S-1}, \tag{5.21}$$

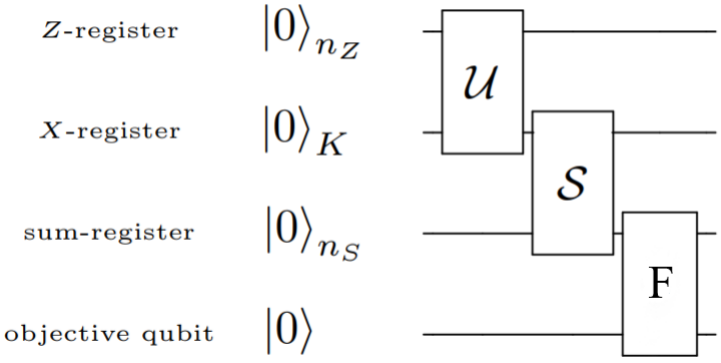


Figura 5.3: Design del circuito di costruzione dell'operatore $\mathcal{A} = F\mathcal{S}\mathcal{U}$. L'objective qubit finale contiene lo stato che fornisce l'output del sistema. La probabilità che venga misurato $|1\rangle$ è la codifica del valore atteso della perdita di portafoglio.

allora è possibile ricavare il valore atteso delle perdite L. Cioè, si ricava $\mathbb{E}\left[\frac{L}{S-1}\right]$, da cui si ottiene direttamente il valore voluto, secondo le proprietà del valore atteso:

$$(S-1)\cdot\mathbb{E}\left[\frac{L}{S-1}\right] = \mathbb{E}\left[L\right]. \tag{5.22}$$

Con questa procedura è anche possibile ricavare gli altri momenti di ordine superiore al primo. Ad esempio, il secondo:

$$\mathbb{E}\left[L^2\right], \tag{5.23}$$

semplicemente implementando

$$f(i) = \frac{i^2}{S-1}. \tag{5.24}$$

Da ciò, segue direttamente che

$$Var(L) = \mathbb{E}\left[L^2\right] - \mathbb{E}^2\left[L\right]. \tag{5.25}$$

A questo punto, sul circuito raffigurato in Figura 5.3 interviene QAE: infatti, sia definito $\mathcal{A}$ l'operatore indicato dalla (3.54) nel seguente modo:

$$\mathcal{A} = F\mathcal{S}\mathcal{U}, \tag{5.26}$$

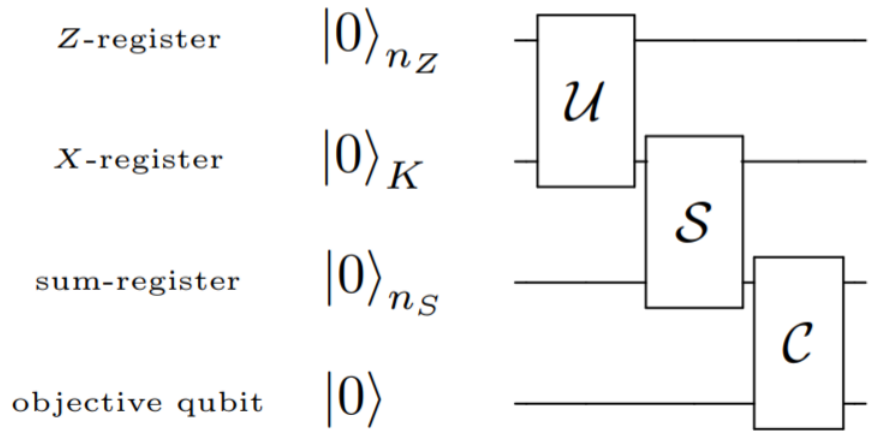


Figura 5.4: Design del circuito di costruzione dell'operatore $\mathcal{A} = \mathcal{CSU}$. L'objective qubit finale contiene lo stato che fornisce l'output del sistema. La probabilità che venga misurato $|1\rangle$ è la codifica di $\mathbb{P}[L \leq l]$, dato un certo l iniziale.

dove F, $\mathcal{U}$ e $\mathcal{S}$ sono date dalle (5.17), (5.6), (5.13), rispettivamente. In altri termini, $\mathcal{A}$ è dato dalla successione di operatori, presente in Figura 5.3, sui $(n_z + K + n_s + 1)$ qubit totali suddivisi in 4 registri. Tramite QAE è possibile fornire una stima di a WHP, che nel caso specifico corrisponde alla probabilità di misurare $|1\rangle$ nello stato (5.18), ossia il valore atteso di L.

5.3 La funzione di distribuzione empirica

In questa sezione, ci si occupa di ricreare su quantum computer uno stato meccanico che descriva la funzione cumulativa di probabilità della perdita totale L. In un contesto classico, ciò si ottiene in maniera analitica solo nei casi più semplici. Quando la dimensione dello spazio di integrazione è troppo grande, allora si ricorre ad una simulazione Monte Carlo.
Ritornando al case di Woerner ed Egger, l'obbiettivo è stimare

$$CDF_L(l) = \mathbb{P}(L \leq l), \tag{5.27}$$

con $l \in Dom(L)$.

Il circuito implementato a tal scopo è riportato in Figura 5.4. Come si osserva, la sua implementazione è identica a quella del circuito descritto nel paragrafo precedente per il calcolo di $\mathbb{E}[L]$, a meno dell'ultimo operatore. F assume la forma di un operatore $\mathcal{C}$, definito *comparator*, che agisce sui medesimi registri, ma con un obbiettivo differente:

$$\mathcal{C} : |i\rangle_{n_s} |0\rangle \rightarrow \begin{cases} |i\rangle_{n_s} |1\rangle & \text{se } i \leq l \\ |i\rangle_{n_s} |0\rangle & \text{altrimenti.} \end{cases}$$

In sostanza, a $\mathcal{C}$ viene associata una funzione $f(i)$ definita nel seguente modo:

$$f(i) = f_l(i) = \begin{cases} 1 & \text{se } i \leq l \\ 0 & \text{se } i \geq l. \end{cases}$$

Lo stato finale degli ultimi due registri, che sono quelli di interesse, risulta essere

$$\mathcal{C}(|\eta\rangle_{n_s} |0\rangle) = \sum_{i=0}^{l} \left(\sqrt{p_i} |i\rangle_{n_s} |1\rangle\right) + \sum_{i=l+1}^{M-1} \left(\sqrt{p_i} |i\rangle_{n_s} |0\rangle\right). \tag{5.28}$$

La probabilità che il measurement dell'ancilla qubit restituisca $|1\rangle$ è uguale a

$$\sum_{i=0}^{l} p_i = \mathbb{P}[L \leq l]. \tag{5.29}$$

Anche in questo caso, interviene il QAE, con

$$\mathcal{A} = \mathcal{CSU}, \tag{5.30}$$

Allo stesso modo di quanto precede, QAE stima WHP la probabilità di misurare $|1\rangle$ sull'ultimo qubit in (5.28), che corrisponde alla CDF di L valutata in l.

5.4 La stima del VaR

Con il circuito costruito fino ad ora, è possibile calcolare le misure di rischio di interesse per il case Credit Risk Analysis. Il VaR è calcolato rispetto ad un orizzonte temporale T, a cui si riferiscono anche le probabilità degli eventi di

default importate sul vettore di stato del QC. Dunque, basta definire un livello di confidenza $1 - \alpha$; dopodiché, vale

$$VaR_{1-\alpha}(L) = \inf\{l \in Dom(L) | \mathbb{P}\left[L > l\right] \leq \alpha\}. \tag{5.31}$$

Indi per cui, per conoscere il valore di $VaR_{1-\alpha}(L) = l_{1-\alpha}$, è necessario utilizzare il circuito implementato per la stima della $CDF(l)$, e poi applicare un algoritmo di bisezione che cerchi il punto inferiore $l_{1-\alpha}$ tale che soddisfi la (5.31). Il costo computazionale che comporta quest'ultimo può essere trascurato rispetto a quello di campionamento quantistico.

Capitolo 6

Un esempio numerico

In questo capitolo viene presentata la fase sperimentale del lavoro di tesi. Innanzitutto, bisogna capire quali sono i passi con cui è stato affrontato il caso numerico e come si arriva alle conclusioni finali.
Il capitolo è così strutturato:

1. si mostra un esempio numerico di Credit Risk Analysis, il quale viene modellato tramite ASRF. Quindi, viene eseguita una simulazione Monte Carlo classica per ottenere una stima precisa delle statistiche di interesse;

2. i risultati precedentemente ottenuti sono posti a confronto con quelli derivanti da una esecuzione di QAE su circuito quantistico;

3. si analizza l'errore quantistico che condiziona le stime. Dopodiché, si presentano i metodi odierni che sono utilizzati per rimediare a tale circostanza.

6.1 L'esecuzione tramite Statevector simulator

La distribuzione delle perdite del portafoglio, modellato tramite ASRF, raggiunge una distribuzione limite unimodale, dovuta al carattere asintotico del portafoglio. L'ipotesi iniziale, necessaria perché tale condizione sia replicata, è che il numero degli assets tenda ad infinito, ma che a livello pratico si traduce in un numero intero sufficientemente elevato.

ASSET	p_0	ρ	LGD
1	0.15	0.05	1
2	0.20	0.1	3

Tabella 6.1: Parametri degli assets di portafoglio.

Tuttavia, questa ipotesi non è rispettata per il caso pratico di cui si vuole effettuare un'analisi. Infatti, bisogna considerare che l'hardware quantistico attuale non consente di trattare problemi di dimensione reale, in quanto i registri circuitali non riescono ad operare su un numero abbastanza elevato di qubits. In questo modo, gli stati totali di misurazione del sistema risultano in numero troppo poco elevato per poter eseguire un'approssimazione del problema in modo efficiente. Pertanto, in attesa che le tecnologie quantistiche siano abbondantemente migliorate, è comunque utile eseguire esperimenti numerici per verificare il funzionamento stimato per via prettamente teorica.
Quindi, sia definito un portafoglio di due soli assets. Il carattere asintotico della distribuzione di perdita non è rispettato, ma questo deriva dalla necessità di analizzare un caso reale di implementazione su quantum computer. Le motivazioni sono le seguenti:

- l'obbiettivo dell'analisi non è dimostrare il carattere analitico delle distribuzioni di perdita del portafoglio asintotico. La tesi non verte infatti sul modello ASRF, bensì sulla sua implementazione su hardware quantistico. Pertanto, al fine di formulare precise considerazioni sul suo funzionamento, è opportuno riportarsi ad un caso più semplice;

- seconda motivazione, non di minor rilievo, riguarda le difficoltà che sorgono per implementare un caso complesso su quantum computer. Più il numero di qubits utilizzati è elevato, maggiore è l'accuratezza con cui è approssimato il problema reale. Tuttavia, come si vedrà nel prosieguo del capitolo, lo stato dell'arte dei circuiti quantistici è ancora lontano dal permettere l'implementazione di circuiti ad un numero adatto di qubits.

I dati che sono utilizzati per i due asset sono risportati in Tabella 6.1.
La simulazione Monte Carlo classica è eseguita tramite un numero di simulazioni

$E[L]$	0.757
$VaR_{0.95}[L]$	3.0
$CVaR_{0.95}[L]$	3.706

Tabella 6.2: Statistiche calcolate sulla distribuzione delle perdite di portafoglio.

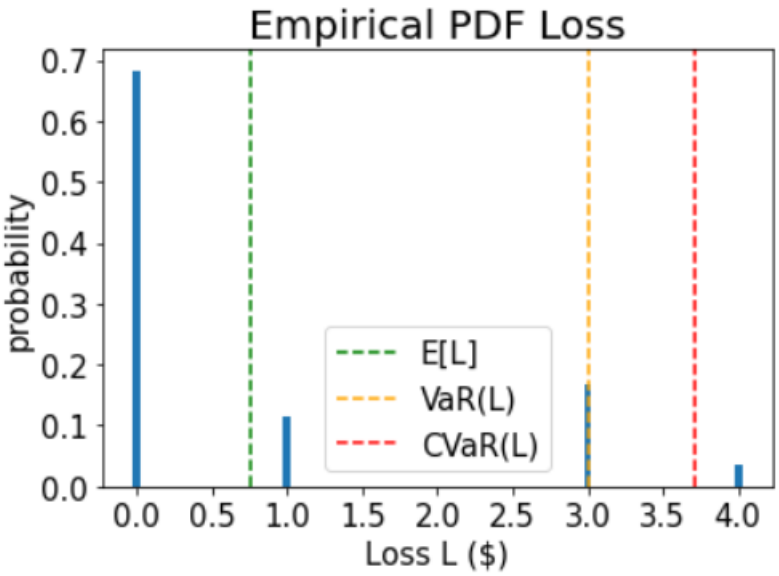


Figura 6.1: PDF empirica della perdita di portafoglio calcolata con MC classico.

M tale da poter garantire un errore di precisione che risulti trascurabile. Sapendo che esso è dell'ordine di $\mathcal{O}(\frac{1}{\sqrt{M}})$, allora è ragionevole che il numero di campioni venga settato a $M = 100000$. In questo modo la stima delle statistiche di interesse possono ritenersi esatte. La funzione di distribuzione empirica delle perdite è ovviamente una funzione discreta, come lo è anche quella teorica per definizione: essa assume valori diversi da zero in corrispondenza di 4 punti precisi del dominio totale. Le statistiche sono state riportate in Tabella 6.2: il livello di confidenza con cui sono calcolati VaR e $CVaR$ è uguale al 95%, per cui vale $\alpha = 0.05$. Siccome si tratta di un caso banale, non sarebbe stato utile fornire un livello α tipico dei casi reali, cioè $\alpha = 0.01$, in quanto entrambe le misure di rischio sarebbero sovrapposte in corrispondenza dell'ultimo quantile della distribuzione. Dunque, per analizzare le performance dell'algoritmo quantistico, è opportuno abbassare il livello di confidenza ad un valore che permetta di distinguere VaR e $CVaR$.

n_z	3
n_s	4
z_{max}	2
m	4

Tabella 6.3: Parametri di circuito.

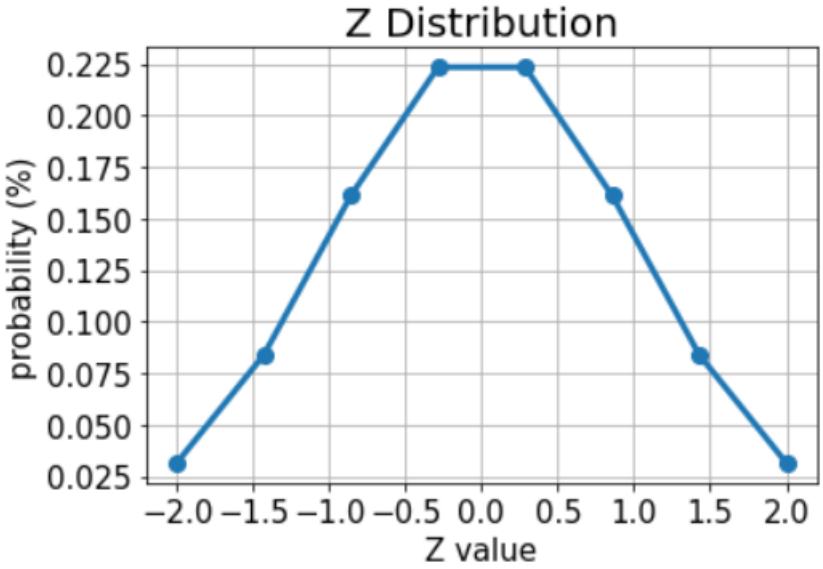


Figura 6.2: Campionamento della variabile latente Z, eseguito su punti equidistanti della sua distribuzione all'interno della regione di massima densità ugule all'intervallo arbitrario $[-z_{max}, +z_{max}]$.

In Figura 6.1 è riportata la distribuzione empirica a seguito della simulazione Monte Carlo classica.

6.2 L'implementazione su circuito e QAE

Lo stesso caso pratico viene affrontato tramite l'utilizzo dell'algoritmo quantistico, che è stato ampiamente analizzato nel Capitolo 5, il cui codice si trova in Appendice *A*.1.

I parametri di portafoglio sono i medesimi della simulazione Monte Carlo classica effettuata nella sezione precedente. Oltre a tali valori, è opportuno definire gli altri

$E[L]$	0.764
$VaR_{0.95}[L]$	3

Tabella 6.4: Statistiche calcolate tramite Statevector simulator.

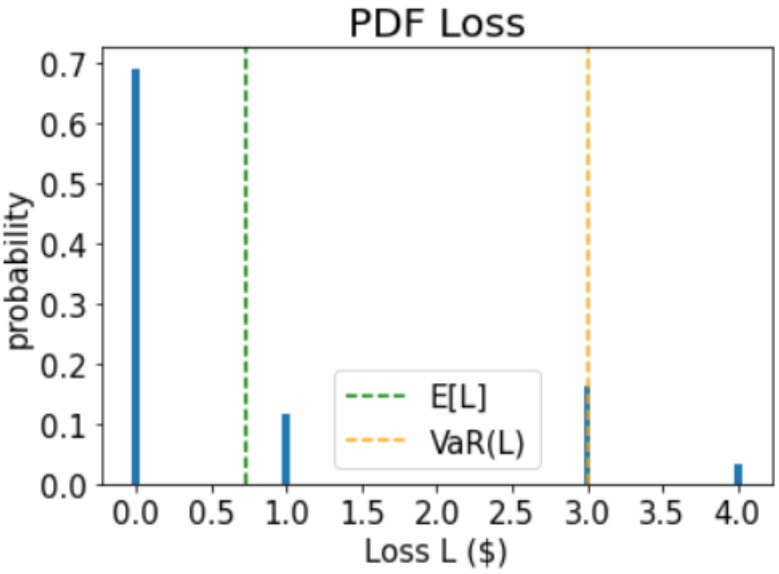


Figura 6.3: PDF empirica della perdita di portafoglio calcolata con Statevector.

parametri che permettono la costruzione fisica del circuito quantistico. Sapendo che la versione reale di QC con il massimo numero di qubits rilasciata da IBM nel suo cloud è composta da 16 elementi, questo numero definisce il limite delle grandezze dei registri utilizzati dall'algoritmo. Tali parametri sono riportati in Tabella 6.3. Il numero di qubits del circuito di implementazione che serve per rappresentare la variabile latente è $n_z = 3$, con cui è possibile ottenere 8 possibili realizzazioni di Z. Queste sono campionate all'interno della regione di dominio di realizzazione della PDF di una variabile normale standard, cioè $\mathbb{R}$, privata di entrambe le code estreme (Figura 6.2). Inoltre, il numero di qubits utilizzato per il *sum*-register è $n_s = 4$, ricavato grazie alla (5.12).

In una prima fase, viene sperimentata la correttezza del circuito. Utilizzando il simulatore Statevector, è possibile rappresentare tutte le realizzazioni di misurazione del vettore di stato, ciascuna delle quali è posta in relazione univoca con

una realizzazione della distribuzione delle perdite L e alla sua probabilità. La distribuzione empirica che si ottiene è riportata in Figura 6.3. I risultati delle statistiche sono riportati in Tabella 6.4. Come si può evincere da un confronto con le stime del precedente paragrafo, il valore atteso delle perdite ottenuto tramite un'esecuzione deterministica del circuito differisce dal valore esatto di un errore dell'ordine di 10^{-2}. Questo è riconducibile alla procedura di discretizzazione della PDF con cui viene trattato il campionamento di Z. Inoltre, il VaR risulta identico alla stima esatta. Tuttavia si presume che lo stesso errore di approssimazione sia conservato all'interno del circuito e che il non verificarsi del medesimo sia giustificato dal fatto che la CDF della perdita totale L è data da una funzione empirica discreta. In altre parole, se si potesse ricostruire un caso meno banale di portafoglio, anche l'errore sul VaR sarebbe visibile.

Nella seconda fase del codice, viene parallelamente condotta la sperimentazione dell'algoritmo QAE. L'obbiettivo è stimare la probabilità che nell'ultimo qubit del circuito venga misurato lo stato $|1\rangle$, il quale ha una probabilità di realizzazione a. In generale, lo stato totale del circuito è definito dalla (3.54).
L'applicazione di QAE, per tutti i casi di utilizzo, viene effettuata con un numero di sampling qubits uguale a $m = 4$. Questo significa che il numero di campionamenti possibili è uguale a $M = 2^m = 16$.

I risultati ottenuti tramite questa simulazione evidenziano quanto l'apporto quantistico sia ancora lontano dall'obbiettivo che si vuole conseguire in ambito di simulazione. Infatti, l'algoritmo QAE, in relazione al corrente caso, fallisce ampiamente la stima delle statistiche di interesse. A titolo di esempio, la distribuzione probabilistica della stima del valore atteso $\hat{a} = \mathbb{E}[L]$ WHP, viene riportata in Figura 6.4. Inoltre, anche la stima di Value at Risk risulta errata con una distorsione molto elevata, tale da invalidare completamente il risultato. Infatti, il valore ottenuto è $VaR_{0.95} = 0$, a fronte di un valore esatto uguale a 3. Tuttavia, questo risultato è ottenuto in seguito ad una elaborazione di dati ricevuti in input anch'essi distorti, provocando un'amplificazione dell'errore.

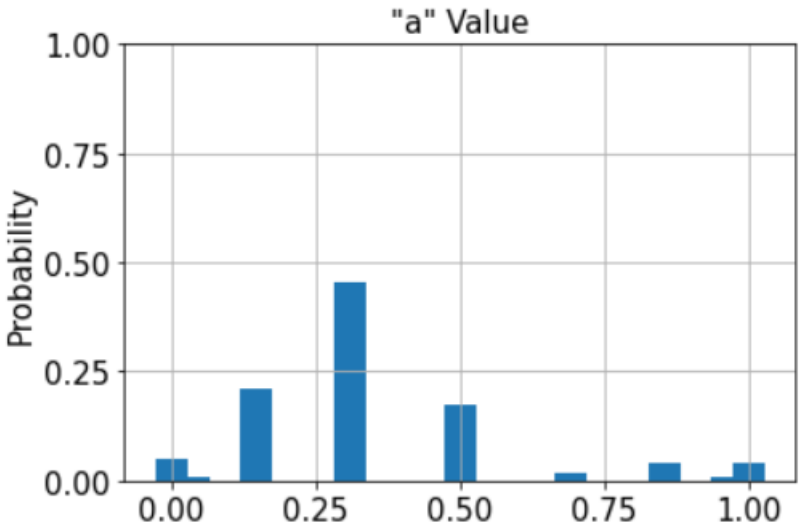


Figura 6.4: Distribuzione di stima del valore atteso $\mathbb{E}[L]$.

Le cause di questa performance pongono le proprie radici su problemi di natura differente, ma in stretta relazione fra loro. Innanzitutto, una frazione dell'errore viene fatta risalire all'implementazione degli operatori: essi approssimano le grandezze reali tramite calcoli che producono stime numeriche a loro volta influenzate da distorsioni di natura operativa. Come già l'intuizione può suggerire, il numero di qubits che viene utilizzato per la costruzione di tali operatori è direttamente proporzionale alla precisione che essi provocano sull'output. In questo modo, verrebbe naturale aumentare la cardinalità dei registri in maniera tale per cui l'errore di approssimazione venga pressoché annullato. Tuttavia, un numero elevato di qubits provoca l'aumento della profondità del circuito e del numero di porte quantistiche necessarie per la manipolazione del vettore di stato. Di conseguenza, come è stato in parte accennato nei precedenti capitoli, aumenta l'errore di decoerenza tipico dell'hardware quantistico di cui si fa utilizzo.
Pertanto, il numero di qubits a cui si fa riscorso costituisce il trade-off che esiste tra errore di approssimazione ed errore di decoerenza quantistica.
La quantificazione di errore che deriva da una e dell'altra fonte non è direttamente osservabile, se eseguita dall'analisi sui risultati. Infatti, la derivazione di ipotesi di errore derivante solamente dall'implementazione degli operatori richiede precise abilità di programmazione quantistica di basso livello. Inoltre, sommati a queste

due cause, coesiste l'errore di amplificazione che è tipico dell'algoritmo QAE: come è stato ricavato per via teorica, il limite inferiore di probabilità tale per cui QAE restituisce un esito corretto è uguale a circa l'81%.

6.3 Le nuove soluzioni: MLAE e IAE

L'imperfezione dell'hardware quantistico e il limite probabilistico dell'algoritmo QAE costituiscono lo scoglio più impervio per la validazione pratica dell'algoritmo di superselezione. Ciononostante, ad ora, esistono delle tecniche quantistiche che tentano di ovviare a questo problema [19]. Basti pensare, infatti, che una buona parte di errore è dovuta alla mancanza di una fase di post-processing in seguito alla stima di a. Quest'ultima equivale ad un punto della griglia definita da

$$\{sin^2\Big(\frac{y\pi}{M}\Big)\}, \tag{6.1}$$

dove $y = 0, \ldots, M/2$.
Le tecniche a cui si fa riferimento sono estensioni di QAE: esse producono stime ricavate da un procedimento quasi del tutto analogo, ma con l'aggiunta di una fase finale di massima verosimiglianza. Per questa ragione, l'algoritmo che ne deriva si chiama *Maximum Likelihood Amplitude Estimation*, grazie al quale vengono prodotti adeguati intervalli di confidenza delle stime. Questo metodo farebbe pensare all'utilizzo di una procedura classica per dover far fronte ad un problema di natura quantistica. Invece, quello che rende notevoli queste estensioni è dovuto al fatto per cui l'iter di massima verosimiglianza è raggiunto da un algoritmo quantistico e non classico. Nella fattispecie, MLAE elimina l'operazione di trasformazione di Fourier inversa per la stima di fase e la sostituisce con la implementazione di nuovi circuiti meno profondi. In modo molto simile è costruito *Iterative Amplitude Estimation*, che costituisce una seconda estensione di QAE. Entrambi riescono a mantenere lo speedup quadratico impartito da QAE, ma allo stesso tempo possiedono performance migliori: essi producono stime sulla base di una fase di post-processing e, soprattutto, esigono una implementazione circuitale meno profonda.

Capitolo 7

Conclusioni

Nella prima parte di questa tesi, è stato ampiamente analizzato l'innovativo metodo computazionale del Quantum Computing. Innanzitutto, è importante definirne l'unità di informazione, ossia il qubit, con cui si codificano i dati reali più semplici. Dopodiché, è stata presentata la teoria relativa alla fisica meccanico-quantistica ed il modo in cui essa viene sfruttata dall'hardware reale per il suo funzionamento. A questo punto, è stato possibile generalizzare i concetti affrontati per sistemi a dimensione maggiore, cioè composti da più qubits in relazione di entanglement. Sono stati presentati i fenomeni di superposition e measurement, che rivestono un ruolo centrale in questo campo scientifico. Inoltre, figurano esempi di porte quantistiche, con cui si può manipolare a piacere il sistema e far ruotare lo stato dei qubits all'interno della Sfera di Bloch, la quale ne fornisce una rappresentazione geometrica.
Per poter compiere la ricerca inerente a questo elaborato, è opportuno effettuare una fase iniziale di studio del Quantum Computing, a cui ha fatto seguito una sperimentazione di circuiti quantistici su simulatori e hardware reale. Dunque, nella fase conclusiva del capitolo iniziale, viene fatto riferimento agli strumenti utilizzati: Qiskit è una piattaforma open-source, messa a disposizione da IBM sul proprio sito online. Essa è ricca di tutorials e metodi utili per affrontare le ricerche in questo ambito. Inoltre, essa permette di implementare i circuiti quantistici su cui sono eseguite le trasformazioni lineari del vettore di stato.

In seguito, si è aperta un'approfondita trattazione teorica degli algoritmi quantistici. Quello formulato da Grover serve per risolvere un problema di ricerca. Esso permette di introdurre il concetto di quantum speed-up, tipico anche di altri algoritmi e della computazione quantistica in generale. Infatti, come si è visto, l'algoritmo di Grover offre la dimostrazione teorica tale per cui si può asserire che un calcolatore quantistico può risolvere un problema matematico in un tempo che è di un ordine quadraticamente inferiore rispetto ad un corrispettivo classico. Grover raggiunge tale risultato tramite un processo di amplificazione delle amplitudes associati agli autostati di misurazione di interesse. E cioè quegli autostati della base computazionale che costituiscono la codifica dei risultati di interesse. Grover funge da esempio illuminante per il seguito dello sviluppo della teoria computazionale quantistica. L'algoritmo su cui verte questo lavoro di tesi è Quantum Amplitude Estimation. Esso prende spunto da Grover, e mantiene il carattere di speed-up quadratico, come Brassard, Hoyer, Mosca e Tapp hanno saputo dimostrare in [13]. QAE, eseguito in relazione ad un preciso vettore di superposition, permette di stimare il valore che possiede una delle sue amplitude, che non sono mai osservabili direttamente. In questo modo, si arriva ad asserire che QAE fornisce un'alternativa efficiente ai metodi Monte Carlo classici per la stima dei risultati del problema matematico preso in esame.

Questo lavoro di tesi è servito ad illustrare come il Quantum Computing può fornire un valido strumento di calcolo per applicazioni in ambito di Credit Risk Analysis. Ovviamente, come si è visto, i rami scientifici, su cui esso può trovare una valido impiego, riguardano svariati ambiti, non solo di natura economica. Infatti, tale scoperta ha avuto ampi risolti per la ricerca nel settore chimico, ma non solo. Anche nella risoluzione di problemi di ottimizzazione e di simulazione di generico ramo di appartenenza, basta essere in grado di formulare il problema in modo corretto nelle sue specifiche di base. Tuttavia, l'analisi dell'elaborato è ristretta ad un problema di Risk Management. Nella fattispecie, è stato descritto uno dei metodi con cui gli istituti di credito devono fare fronte agli eventuali defaults dei loro debitori. Tali posizioni di rischio sono contenute in un portafoglio sottoforma di assets economici. Secondo gli Accordi di Basilea, uno dei metodi con cui si deve monitorare questo oggetto economico è il modello Asympotic Single Risk Factor.

Tramite questo strumento, è possibile stimare la distribuzione delle perdite ad un orizzonte temporale breve prefissato ed è calcolata sull'intero portafoglio. In questo modo, è possibile risalire alle statistiche di interesse, come il valore atteso e, soprattutto, il Value at Risk, con cui si può stimare il capitale minimo di liquidità. Dopo aver introdotto anche il problema di rischio secondo una procedura analitica, il prosieguo ha esaminato il modo con cui il modello ASRF è codificato sui circuiti quantistici. Per questa parte è stato seguito un caso di studio presente in Qiskit, presentato in [2]. Una volta caricata l'informazione sull'hardware quantistico, è possibile stimare le statistiche di base con l'ausilio di QAE.
L'esempio con cui sono stati sperimentati i circuiti e gli algoritmi quantistici è assai lontano dall'essere considerato un problema di natura reale. Ciò è dovuto alle sue dimensioni, decisamente ridotte: infatti, se un portafoglio azionario può arrivare a contare anche migliaia di assets, in questo caso il numero di crediti a rischio è ridotto a soli due. Inoltre, i dati con cui sono settati i parametri del portafoglio e del modello non derivano da un'analisi di mercato reale. Basti pensare, ad esempio, che l'implementazione su circuiti quantistici non permette per alcuni di essi la codifica di valori reali. Indi per cui, è necessario giungere ad una approssimazione intera. Ciononostante, è importante ricordare che l'obbiettivo di questa tesi non è incentrato nè sullo studio del modello ASRF, nè tantomeno sul suo comportamento in un caso reale. Questa situazione, che vede la simulazione per un esempio pratico apparentemente banale e semplicistico, serve ad evidenziare qual è l'attuale stato dell'arte del Quantum Computing e quanta strada ci sia ancora da percorrere per poter effettivamente rappresentare uno strumento di calcolo, non solo efficiente, ma anche efficace. A supporto di tale tesi intervengono i risultati ottenuti dall'esecuzione dei circuiti quantistici e quelli che derivano da una simulazione Monte Carlo classica. Tramite l'uso del simulatore Statevector è stata dimostrata la corretta codifica del problema sul circuito. Infatti le statistiche sulla distribuzione di perdita totale, valore atteso e Value at Risk, coincidono a meno di un errore dell'ordine di 10^{-2}. Invece, l'esecuzione su hardware reale combinato a Quantum Amplitude Estimation non consente di giungere al risultato corretto. Indagare sull'errore dell'output non è immediato, perché le fonti da cui scaturisce sono molteplici e di diversa natura, ma contribuiscono contemporaneamente nella

sua formazione. Bisogna tenere presente che il numero di qubits che sono utilizzati per l'implementazione dei circuiti contribuisce alla precisione dell'output. Maggiore è la cardinalità dei registri, minore sarà l'errore di approssimazione con cui si simula il problema reale e maggiore sarà la precisione dell'output. Inoltre, come si è visto, l'algoritmo QAE fa utilizzo di campionamenti quantistici, il cui numero è dato in relazione esponenziale dal numero dei counting qubits. Tuttavia, l'hardware reale, che rientra tra quelli messi a disposizione di IBM, fa affidamento ad un processore quantistico che conta solamente 16 qubits. Quindi, un numero pressochè esiguo per poter trattare un problema di dimensione reale. Inoltre, a questo si aggiunge anche l'imprecisione di cui è affetta la macchina quantistica. L'epoca in cui ci apprestiamo a vivere è entrata nell'immaginario scientifico collettivo come una fase di transizione, definita NISQ era. Gli hardware reali quantistici non sono macchine perfette. Essi soffrono di un errore di fondo che è causato dalla difficoltà di progettazione fisica dei computer quantistici. Infatti, perché essi funzionino in maniera efficiente, è necessario che il sistema fisico-meccanico su cui si basano costituiscano un sistema totalmente isolato dalle interferenze esterne. Da un punto di vista progettuale questa è un'impresa non di poco conto. Per questo motivo l'obbiettivo dichiarato circa la nascita dei futuri processori quantistici riguarda l'aumento del numero di qubits, ma soprattutto il miglioramento della qualità durante l'esecuzione.

In definitiva, ad oggi, la parte teorica sviluppata negli ultimi decenni sta cercando un riscontro pratico tramite l'esecuzione degli algoritmi citati su hardware reale. Tuttavia, quest'ultimo non garantisce ancora l'efficienza necessaria per il fine desiderato. L'errore di decoerenza e la cardinalità ridotta dei registri determinano il collo di bottiglia dell'intera analisi.

Appendice A

Source Code

A.1 Quantum Amplitude Estimation

```
from qiskit import QuantumRegister, QuantumCircuit, BasicAer, execute

from qiskit.aqua.components.uncertainty_models import
    GaussianConditionalIndependenceModel as GCI
from qiskit.aqua.components.uncertainty_problems import
    UnivariatePiecewiseLinearObjective as PwlObjective
from qiskit.aqua.components.uncertainty_problems import
    MultivariateProblem
from qiskit.aqua.circuits import WeightedSumOperator
from qiskit.aqua.circuits  import FixedValueComparator as Comparator
from qiskit.aqua.algorithms import AmplitudeEstimation

import numpy as np
import matplotlib.pyplot as plt
```

```
# define backend to be used
backend = BasicAer.get_backend('statevector_simulator')
```

```
# set problem parameters
n_z = 3  #originale=2
z_max = 2
z_values = np.linspace(-z_max, z_max, 2**n_z)
p_zeros = [0.15, 0.20]
rhos = [0.05, 0.1]
lgd = [1, 3]
K = len(p_zeros)
alpha = 0.05
```

```
# construct circuit factory for uncertainty model (Gaussian
    Conditional Independence model)
u = GCI(n_z, z_max, p_zeros, rhos)
# determine the number of qubits required to represent the
    uncertainty model
num_qubits = u.num_target_qubits

# initialize quantum register and circuit
q = QuantumRegister(num_qubits, name='q')
qc = QuantumCircuit(q)

# construct circuit
u.build(qc, q)
# run the circuit and analyze the results
job = execute(qc, backend=BasicAer.get_backend('statevector_simulator
    '))

# analyze uncertainty circuit and determine exact solutions
p_z = np.zeros(2**n_z)
p_default = np.zeros(K)
values = []
probabilities = []
for i, a in enumerate(job.result().get_statevector()):

    # get binary representation
```

```
23      b = ('{0:0%sb}' % num_qubits).format(i)
24      prob = np.abs(a)**2
25
26      # extract value of Z and corresponding probability
27      i_normal = int(b[-n_z:], 2)
28      p_z[i_normal] += prob
29
30      # determine overall default probability for k
31      loss = 0
32      for k in range(K):
33          if b[K - k - 1] == '1':
34              p_default[k] += prob
35              loss += lgd[k]
36      values += [loss]
37      probabilities += [prob]
38
39  values = np.array(values)
40  probabilities = np.array(probabilities)
41
42  expected_loss = np.dot(values, probabilities)
43
44  losses = np.sort(np.unique(values))
45  pdf = np.zeros(len(losses))
46  for i, v in enumerate(losses):
47      pdf[i] += sum(probabilities[values == v])
48  cdf = np.cumsum(pdf)
49
50  i_var = np.argmax(cdf >= 1-alpha)
51  exact_var = losses[i_var]
52  exact_cvar = np.dot(pdf[(i_var+1):], losses[(i_var+1):])/sum(pdf[(
        i_var+1):])
53  # exact_cvar = np.dot(pdf[(i_var+1):], losses[(i_var+1):])/alpha
54
55  print('Expected Loss E[L]:                %.4f' % expected_loss)
56  print('Value at Risk VaR[L]:              %.4f' % exact_var)
57  print('P[L <= VaR[L]]:                    %.4f' % cdf[exact_var])
58  #print('Conditional Value at Risk CVaR[L]: %.4f' % exact_cvar)
59
```

```
60 # plot loss PDF, expected loss, var, and cvar
61 plt.bar(losses, pdf, width=0.05)
62 plt.axvline(expected_loss, color='green', linestyle='--', label='E[L]
       ')
63 plt.axvline(exact_var, color='orange', linestyle='--', label='VaR(L)'
       )
64 #plt.axvline(exact_cvar, color='red', linestyle='--', label='CVaR(L)
       ')
65 plt.legend(fontsize=15)
66 plt.xlabel('Loss L ($)', size=15)
67 plt.ylabel('probability', size=15)
68 plt.title('PDF Loss', size=20)
69 plt.xticks(size=15)
70 plt.yticks(size=15)
71 plt.show()
72
73 # plot results for Z
74 plt.plot(z_values, p_z, 'o-', linewidth=3, markersize=8)
75 plt.grid()
76 plt.xlabel('Z value', size=15)
77 plt.ylabel('probability (%)', size=15)
78 plt.title('Z Distribution', size=20)
79 plt.xticks(size=15)
80 plt.yticks(size=15)
81 plt.show()
82
83 # plot results for default probabilities
84 plt.bar(range(K), p_default)
85 plt.xlabel('Asset', size=15)
86 plt.ylabel('probability (%)', size=15)
87 plt.title('Individual Default Probabilities', size=20)
88 plt.xticks(range(K), size=15)
89 plt.yticks(size=15)
90 plt.grid()
91 plt.show()
```

```
# determine number of qubits required to represent total loss
n_s = WeightedSumOperator.get_required_sum_qubits(lgd)

# create circuit factory (add Z qubits with weight/loss 0)
agg = WeightedSumOperator(n_z + K, [0]*n_z + lgd)

# define linear objective function
breakpoints = [0]
slopes = [1]
offsets = [0]
f_min = 0
f_max = sum(lgd)
c_approx = 0.25

objective = PwlObjective(
    agg.num_sum_qubits,
    0,
    2**agg.num_sum_qubits-1,  # max value that can be reached by the
    qubit register
    breakpoints,
    slopes,
    offsets,
    f_min,
    f_max,
    c_approx
)

# define overall multivariate problem
multivariate = MultivariateProblem(u, agg, objective)

num_qubits = multivariate.num_target_qubits
num_ancillas = multivariate.required_ancillas()

q = QuantumRegister(num_qubits, name='q')
q_a = QuantumRegister(num_ancillas, name='q_a')
qc = QuantumCircuit(q, q_a)

multivariate.build(qc, q, q_a)
```

```
38
39 qc.draw()
40
41 job = execute(qc, backend=BasicAer.get_backend('statevector_simulator
       '))
42
43 # evaluate resulting statevector
44 value = 0
45
46 for i, a in enumerate(job.result().get_statevector()):
47     multivariate.num_target_qubits).format(i)[-multivariate.
       num_target_qubits:]
48     am = np.round(np.real(a), decimals=4)
49     if np.abs(am) > 1e-6 and b[0] == '1':
50         value += am**2
51
52 print('Exact Expected Loss:   %.4f' % expected_loss)
53 print('Exact Operator Value:  %.4f' % value)
54 print('Mapped Operator value: %.4f' % multivariate.
       value_to_estimation(value))
55
56 # run amplitude estimation
57 num_eval_qubits = 4
58 shots = 100
59 ae = AmplitudeEstimation(num_eval_qubits, multivariate)
60 result = ae.run(quantum_instance=BasicAer.get_backend('qasm_simulator
       '), shots=shots)
61 # result = ae.run(quantum_instance=BasicAer.get_backend('
       statevector_simulator'))
62
63 # print results
64 print('Exact value:    \t%.4f' % expected_loss)
65 print('Estimated value:\t%.4f' % result['estimation'])
66 print('Probability:    \t%.4f' % result['max_probability'])
67
68 # plot estimated values for "a"
69 plt.bar(result['values'], result['probabilities'], width=0.5/len(
       result['probabilities']))
```

```
plt.xticks([0, 0.25, 0.5, 0.75, 1], size=15)
plt.yticks([0, 0.25, 0.5, 0.75, 1], size=15)
plt.title('"a" Value', size=15)
plt.ylabel('Probability', size=15)
plt.ylim((0,1))
plt.grid()
plt.show()

# plot estimated values for expected loss (after re-scaling and reversing the c_approx-transformation)
plt.bar(result['mapped_values'], result['probabilities'], width=0.5/len(result['probabilities']))
plt.axvline(expected_loss, color='red', linestyle='--', linewidth=2)
plt.xticks(size=15)
plt.yticks([0, 0.25, 0.5, 0.75, 1], size=15)
plt.title('Expected Loss', size=15)
plt.ylabel('Probability', size=15)
plt.ylim((0,1))
plt.grid()
plt.show()
```

```

# define value x to evaluate the CDF(x)
def get_cdf_operator_factory(x_eval):

    # comparator as objective
    cdf_objective = Comparator(agg.num_sum_qubits, x_eval+1, geq=False)

    # define overall uncertainty problem
    multivariate_cdf = MultivariateProblem(u, agg, cdf_objective)

    return multivariate_cdf

# set x value to estimate the CDF
x_eval = 3
```

```
15
16 # get operator
17 multivariate_cdf = get_cdf_operator_factory(x_eval)
18
19 # get required number of qubits
20 num_qubits = multivariate_cdf.num_target_qubits
21 num_ancillas = multivariate_cdf.required_ancillas()
22
23 # construct circuit
24 q = QuantumRegister(num_qubits, name='q')
25 q_a = QuantumRegister(num_ancillas, name='q_a')
26 qc = QuantumCircuit(q, q_a)
27
28 multivariate_cdf.build(qc, q, q_a)
29
30 job = execute(qc, backend=BasicAer.get_backend('statevector_simulator
    '))
31
32 qc.draw()
33
34 # evaluate resulting statevector
35 var_prob = 0
36 for i, a in enumerate(job.result().get_statevector()):
37     b = ('{0:0%sb}' % multivariate_cdf.num_target_qubits).format(i)[-
    multivariate_cdf.num_target_qubits:]
38     prob = np.abs(a)**2
39     if prob > 1e-6 and b[0] == '1':
40         var_prob += prob
41 print('Operator CDF(%s)' % x_eval + ' = %.4f' % var_prob)
42 print('Exact    CDF(%s)' % x_eval + ' = %.4f' % cdf[x_eval])
43
44 # run amplitude estimation
45 #num_eval_qubits = 4
46 ae_cdf = AmplitudeEstimation(num_eval_qubits, multivariate_cdf)
47 result_cdf = ae_cdf.run(quantum_instance=BasicAer.get_backend('
    qasm_simulator'), shots=shots)
48 #result_cdf = ae_cdf.run(quantum_instance=BasicAer.get_backend('
    statevector_simulator'))
```

```
49
50 # print results
51 print('Exact value:    \t%.4f' % cdf[x_eval])
52 print('Estimated value:\t%.4f' % result_cdf['estimation'])
53 print('Probability:    \t%.4f' % result_cdf['max_probability'])
54
55 # plot estimated values for "a"
56 plt.bar(result_cdf['values'], result_cdf['probabilities'], width=0.5/
       len(result['probabilities']))
57 plt.axvline(cdf[x_eval], color='red', linestyle='--', linewidth=2)
58 plt.xticks([0, 0.25, 0.5, 0.75, 1], size=15)
59 plt.yticks([0, 0.25, 0.5, 0.75, 1], size=15)
60 plt.title('CDF(%s)' % x_eval, size=15)
61 plt.ylabel('Probability', size=15)
62 plt.ylim((0,1))
63 plt.grid()
64 plt.show()
```

```
1  def run_ae_for_cdf(x_eval, num_eval_qubits=3, simulator='
       statevector_simulator'):
2
3      # run amplitude estimation
4      multivariate_var = get_cdf_operator_factory(x_eval)
5      ae_var = AmplitudeEstimation(num_eval_qubits, multivariate_var)
6      result_var = ae_var.run(BasicAer.get_backend(simulator))
7
8      return result_var['estimation']
9
10 def bisection_search(objective, target_value, low_level, high_level,
       low_value=None, high_value=None):
11     """
12     Determines the smallest level such that the objective value is
       still larger than the target
13     :param objective: objective function
14     :param target: target value
15     :param low_level: lowest level to be considered
```

```
16      :param high_level: highest level to be considered
17      :param low_value: value of lowest level (will be evaluated if set
        to None)
18      :param high_value: value of highest level (will be evaluated if
        set to None)
19      :return: dictionary with level, value, num_eval
20      """
21
22      # check whether low and high values are given and evaluated them
        otherwise
23      print('
        ------------------------------------------------------------------
        ')
24      print('start bisection search for target value %.3f' %
        target_value)
25      print('
        ------------------------------------------------------------------
        ')
26      num_eval = 0
27      if low_value is None:
28          low_value = objective(low_level)
29          num_eval += 1
30      if high_value is None:
31          high_value = objective(high_level)
32          num_eval += 1
33
34      # check if low_value already satisfies the condition
35      if low_value > target_value:
36          return {'level': low_level, 'value': low_value, 'num_eval':
        num_eval, 'comment': 'returned low value'}
37      elif low_value == target_value:
38          return {'level': low_level, 'value': low_value, 'num_eval':
        num_eval, 'comment': 'success'}
39
40      # check if high_value is above target
41      if high_value < target_value:
42          return {'level': high_level, 'value': high_value, 'num_eval':
         num_eval, 'comment': 'returned low value'}
```

```
43      elif high_value == target_value:
44          return {'level': high_level, 'value': high_value, 'num_eval':
        num_eval, 'comment': 'success'}
45
46      # perform bisection search until
47      print('low_level    low_value    level    value    high_level
        high_value')
48      print('
        ------------------------------------------------------------------
        ')
49      while high_level - low_level > 1:
50
51          level = int(np.round((high_level + low_level) / 2.0))
52          num_eval += 1
53          value = objective(level)
54
55          print('%2d            %.3f        %2d        %.3f     %2d
            %.3f' \
56                % (low_level, low_value, level, value, high_level,
        high_value))
57
58          if value >= target_value:
59              high_level = level
60              high_value = value
61          else:
62              low_level = level
63              low_value = value
64
65      # return high value after bisection search
66      print('
        ------------------------------------------------------------------
        ')
67      print('finished bisection search')
68      print('
        ------------------------------------------------------------------
        ')
69      return {'level': high_level, 'value': high_value, 'num_eval':
        num_eval, 'comment': 'success'}
```

```
70
71 # run bisection search to determine VaR
72 #num_eval_qubits = 4
73 objective = lambda x: run_ae_for_cdf(x, num_eval_qubits=
       num_eval_qubits)
74 bisection_result = bisection_search(objective, 1-alpha, min(losses)
       -1, max(losses), low_value=0, high_value=1)
75 var = bisection_result['level']
76
77 print('Estimated Value at Risk: %2d' % var)
78 print('Exact Value at Risk:     %2d' % exact_var)
79 print('Estimated Probability:    %.3f' % bisection_result['value'])
80 print('Exact Probability:        %.3f' % cdf[exact_var])
```

```
import qiskit.tools.jupyter
%qiskit_version_table
%qiskit_copyright
```

Version Information

Qiskit Software	Version
Qiskit	0.19.3
Terra	0.14.1
Aer	0.5.2
Ignis	0.3.0
Aqua	0.7.1
IBM Q Provider	0.7.2
System information	
Python	3.7.7 (default, May 6 2020, 11:45:54) [MSC v.1916 64 bit (AMD64)]
OS	Windows
CPUs	4
Memory (Gb)	15.772918701171875
	Wed Aug 26 12:19:07 2020 ora legale Europa occidentale

This code is a part of Qiskit

A.2 Simulazione Monte Carlo classica

```
# import all required libreries

import pandas as pd
import numpy as np
import random
from scipy import stats
from scipy.stats import bernoulli
from scipy.stats import norm
import math
import matplotlib.pyplot as plt
```

```
# set problem parameters

p_zeros = [0.15, 0.20]
rhos = [0.05, 0.1]
lgd = [1, 3]
K = len(p_zeros)
alpha = 0.05
num_simulations = 100000
```

```
# function to sample bernoulli variables

def calculate_p(z, p_zeros, rhos):

    p = np.zeros((np.size(z), np.size(p_zeros)))

    for i in range(0, np.size(z)):
        for k in range(0, np.size(p_zeros)):
            p[i,k] = norm.cdf((norm.ppf(p_zeros[k])-np.sqrt(rhos[k])*
    z[i])/np.sqrt(1-rhos[k]))

```

```
    return p
```

```
# function to evaluate losses per each scenario

def calculate_loss_per_scenario(p, lgd):

    losses = np.zeros(p.shape[0])
    for i in range(0, p.shape[0]):
        for j in range(0, np.size(lgd)):
            losses[i] = losses[i] + lgd[j]*bernoulli.rvs(p[i, j])
    return losses
```

```
# function to calculate var

def calculate_var(losses, alpha):

    vect = losses
    vect = np.sort(vect)
    index = math.trunc((1-alpha)*np.size(vect))

    return vect[index]
```

```
# function to calculate cvar

def calculate_cvar(losses, alpha):

    vect = losses
    vect = np.sort(vect)
    index = math.trunc((1-alpha)*np.size(vect))+1
    if (np.size(vect)-index != 0):
        cvar = sum(vect[index:])/(np.size(vect)-index)
    else:
        cvar = vect[-1]
```

```
    return cvar
```

```
# function to estimate empirical PDF and CDF of losses

def calculate_pdf_cdf(array):
    values = np.sort(np.unique(array))
    temp = np.sort(array)
    pdf = np.zeros(np.size(values))

    index = 0
    for i in range(0, np.size(array)):
        if temp[i]==values[index]:
            pdf[index] += 1
        else:
            index += 1
            pdf[index] += 1
    return pdf/np.size(array), np.cumsum(pdf)/np.size(array)
```

```
# main of the code

np.random.seed(30)
z = np.random.standard_normal(num_simulations)
p = calculate_p(z, p_zeros, rhos)
losses = calculate_loss_per_scenario(p, lgd)
expected_loss = sum(losses)/num_simulations
var = calculate_var(losses, alpha)
cvar = calculate_cvar(losses, alpha)
```

```
# plot loss PDF, expected loss, var, and cvar

plt.bar(np.sort(np.unique(losses)), pdf, align = "center", width =
    0.05)
plt.axvline(expected_loss, color='green', linestyle='--', label='E[L]
    ')
```

```
plt.axvline(var, color='orange', linestyle='--', label='VaR(L)')
plt.axvline(cvar, color='red', linestyle='--', label='CVaR(L)')
plt.legend(fontsize=15)
plt.xlabel('Loss L ($)', size=15)
plt.ylabel('probability', size=15)
plt.title('Empirical PDF Loss', size=20)
plt.xticks(size=15)
plt.yticks(size=15)
plt.show()

print('expected loss = ', expected_loss)
print('var = ', var)
print('cvar = ', cvar)
```

Appendice B

Gli operatori

B.1 Weighted Sum Operator

Per l'implementazione del *Weighted Sum Operator* $\mathcal{S}$ si utilizza il seguente procedimento [18].
Computazionalmente $\mathcal{S}$ opera la seguente trasformazione:

$$S\left|a\right\rangle_n\left|0\right\rangle_m = \left|a\right\rangle_n\left|\sum_{i=1}^{n} w_i a_i\right\rangle_m , \tag{B.1}$$

dove $m = \lfloor \log_2 \left(\sum_{i=1}^{n} w_i\right) \rfloor$ garantisce che il risultato più grande della somma possa essere rappresentato. In altre parole, $\mathcal{S}$ riceve in ingresso uno stato $\left|a\right\rangle_n = \left|a_1 \ldots a_n\right\rangle$ di un registro a n qubits, univocamente collegati ai pesi $(w_1, \ldots, w_n)$. Dopodiché, esso restituisce in output su un secondo registro ad m qubits la somma $\left|s\right\rangle_m = \left|s_1 \ldots s_m\right\rangle$.
Si definisca una matrice binaria $\Omega \in \{0,1\}^{n \times n^*}$, in cui ogni riga è la rappresentazione binaria dei pesi w e $n*$ è il numero massimo di cifre utilizzato per tale conversione. In questo modo, $\mathcal{S}$ opera una somma dell'i-esimo qubit $\left|a_1\right\rangle$ del primo registro $\left|a\right\rangle$ al j-esimo qubit $\left|s_j\right\rangle$ del registro della somma $\left|s\right\rangle$ se e solo se $\Omega_{ij} = 1$.
Per questa operazione, spesso è opportuno l'intervento di un secondo registro di ancilla qubits $\left|c_j\right\rangle$ che funge da memoria per la somma di un bit al qubit $\left|s_j\right\rangle$.
L'operatore $\mathcal{S}$ è creato tramite l'utilizzo dei gates X, $CNOT$, $CCNOT$. Iterando per ogni colonna di Ω, a partire dalla prima, si trovi ogni elemento tale per cui

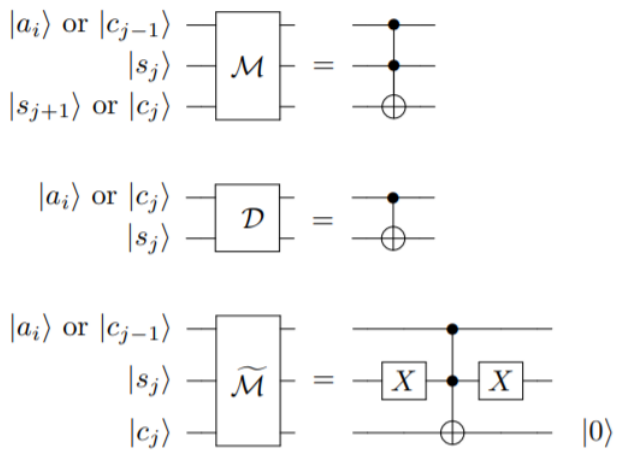


Figura B.1: A partire dall'alto, Toffoli, CNOT e circuito di reset.

$\Omega_{ij} = 1$ e si sommi il corrispondente qubit $|a_i\rangle$ a $|s_j\rangle$. Per la somma di due qubits vengono utilizzati i gates precedentemente citati. Consecutivamente devono essere eseguite le seguenti operazioni:

1. per conservare il risultato temporaneo sul qubit $|c_j\rangle$, si applichi Toffoli a $|a_i\rangle$, $|s_j\rangle$ e $|c_j\rangle$;

2. calcolo del risultato temporaneo tramite $CNOT$. La somma di $|a_i\rangle$ e $|s_j\rangle$ viene conservata in $|c_j\rangle$;

3. iterando sui primi due passaggi, si somma $|c_j\rangle$ al qubit successivo $|s_{j+1}\rangle$, il cui risultato viene conservato in $|c_{j+1}\rangle$;

4. si ripeta il terzo passo fino a quando tutti i qubit $|c_j\rangle$ sono utilizzati;

5. reinizializzazione del qubit $|c_j\rangle$.

Queste operazioni sono riportate in Figura $B.1$. Questo permette di ricreare lo stato finale definito dalla (£B.1).

B.2 Gli operatori di Householder

L'applicazione di QAE richiede l'appropriata implementazione fisica su hardware dei circuiti che rappresentano gli operatori di riflessione. Pertanto, si faccia menzione dell'operatore con cui si esegue il campionamento sul registro di valutazione:

$$\mathcal{Q} = Q(\mathcal{A}, f) = -\mathcal{A}S_0\mathcal{A}^{-1}S_{\psi_0}. \tag{B.2}$$

L'operazione di riflessione S_0 è creata tramite il seguente iter di phase kickback [17]: si inizializza un ancilla qubit in stato $|1\rangle$ tramite X, dopodiché si inserisce un *multi-controlled Z Gate* in cui i controlli sono tutti i qubits su cui agisce $\mathcal{A}$ ed il target è posto sull'ancilla. Infine, quest'ultimo viene nuovamente invertito tramite X. In questo modo il multi-controlled Z agisce solamente se tutti i qubits del registro sono in stato $|0\rangle$.
Tramite una procedura simile, viene costruito il design di S_{ψ_0}. A differenza del precedente, il multi-controlled Z viene sostituito con un controllo semplice solamente sull objective qubit del registro di $\mathcal{A}$. In questo modo l'amplitude dello stato $|1\rangle$ dell'objective qubit viene opposta in segno, comportando la riflessione dovuta da S_{ψ_0}.
Per entrambi S_0 e S_{ψ_0} si può trascurare il loro tempo di esecuzione rispetto a quello totale. Infatti, è stato dimostrato che la loro realizzazione può essere ricavata trmite una serie di gate semplici con una profondità di ordine logaritmico e con un numero lineare di ancillas [20].

Bibliografia

[1] Héctor Abraham et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2019. DOI: 10.5281/zenodo.2562110 (cit. alle pp. 3, 79).

[2] S. Woerner e Daniel j. Egger. «Quantum Risk Analysis». In: (feb. 2019) (cit. alle pp. 4, 79, 101).

[3] P. Dirac. *The Principles of Quantum Mechanics*. Clarendon Press, 1947 (cit. a p. 11).

[4] S. Lloyd. «Quantum Information Science». In: (2009) (cit. a p. 24).

[5] A. Einstein, B. Podolsky e N. Rosen. «Can Quantum-Mechanical Description of Physical Reality be Considered Complete?» In: 47 (mag. 1935), pp. 777–80 (cit. a p. 24).

[6] A. Gilyén, S. Arunachalam e N. Wiebe. «Optimizing quantum optimization algorithms via faster quantum gradient computation». In: (apr. 2018) (cit. a p. 30).

[7] J. Preskill. «Quantum Computing in the NISQ era and beyond». In: (lug. 2018) (cit. a p. 34).

[8] P.W. Shor. «Algorithms for quantum computation: discrete logarithms and factoring». In: (nov. 1994) (cit. a p. 35).

[9] L. Grover e T. Rudolph. «Creating superpositions that correspond to efficiently integrable probability distributions». In: (feb. 2008) (cit. a p. 37).

[10] Lov K. Grover (Bell Labs e Murray Hill NJ). «A fast quantum mechanical algorithm for database search». In: (mag. 1996), p. 8 (cit. a p. 38).

[11] ETH Zurich. «Principles of Distributed Computing (lecture 7)». In: (feb. 2015) (cit. a p. 49).

[12] George F. Viamontes, Igor L. Markov e John P. Hayes). «Is quantum search practical?» In: (apr. 2004) (cit. a p. 51).

[13] G. Brassard, P. Hoyer, M. Mosca e A. Tapp. «Quantum Amplitude Amplification and Estimation». In: (mag. 2000) (cit. alle pp. 52, 53, 64, 100).

[14] M. Nielsen e I. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000 (cit. a p. 59).

[15] Robert C. Merton. «On the pricing of corporate debt: the risk structure of interest rates». In: *Journal of Finance* 29 (lug. 1974), pp. 449–470 (cit. a p. 69).

[16] John C. Hull. *Options, Futures, and Other Derivatives*. Pearson, 2018 (cit. a p. 69).

[17] S. Woerner, D.J. Egger e J. Mestre R. Gutierrez. «Quantum Risk Analysis using Quantum Computers». In: (lug. 2019) (cit. alle pp. 83, 122).

[18] S. Woerner, D.J. Egger, R. Gutierrez, C. Zoufal, R. Iten, N. Shen e Y. Sun and. «Option Pricing using Quantum Computers». In: (lug. 2020) (cit. alle pp. 85, 120).

[19] D. Grinko, J. Gacon, C. Zoufal e S. Woerner. «Iterative Quantum Amplitude Estimation». In: (dic. 2019) (cit. a p. 98).

[20] D. Maslov. «On the advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization». In: (feb. 2016) (cit. a p. 122).