# Informatica quantistica con esempi in linguaggio C

Francesco e Valentina Sisini

1 gennaio 2025

Un prodotto di: Scuola Sisini

https://www.facebook.com/scuolasisini

https://www.scuolasisini.com

Terza edizione 2025

Prima pubblicazione in Italia 2020

*Questo testo è dedicato a Carlo Ferrario (1949-2015), relatore della mia tesi di Laurea. Per la sua lucidità di pensiero e chiarezza espositiva è stato un punto di riferimento per diverse generazioni di studenti e studentesse che si sono formati presso l'università degli studi di Ferrara.*
*Per negligenza, non l'ho ringraziato allora. Lo faccio adesso.*
Scuola Sisini ringrazia calorosamente Lauro Galtarossa per la rilettura, non specializzata ma condotta in forma amichevole, del testo.

# Indice

# Capitolo 1

# Introduzione

## 1.1 Perché imparare l'informatica quantistica

L'informatica si è sviluppata principalmente attraverso le discipline della fisica e della matematica. Da un lato, la matematica ha fornito le linee guida per lo sviluppo del calcolo formale; dall'altro, la fisica ha contribuito con lo sviluppo delle tecnologie che hanno permesso l'implementazione concreta dei meccanismi di calcolo automatico, come le valvole termoioniche e i transistor. Il legame tra queste tre discipline è forte sin dalle loro origini, ma fino alla metà del XX secolo esse hanno sostanzialmente mantenuto una propria identità disciplinare. In pratica, i principi ontologici della fisica erano sostanzialmente indipendenti da quelli dell'informatica e della matematica, e viceversa.

A partire dalla metà del XIX secolo, il paradigma fisico si è spostato dal meccanicismo al paradigma energetico, in cui l'universo veniva visto come un insieme di trasformazioni che conservano l'energia e aumentano l'entropia, secondo i principi della termodinamica. La meccanica quantistica, sviluppatasi all'inizio del XX secolo, ha consolidato questa visione, ma ha anche introdotto nuove prospettive. Ha mostrato che nel mondo microscopico i principi di conservazione dell'energia possono essere violati su scale temporali estremamente brevi, in accordo con il principio di indeterminazione di Heisenberg e le fluttuazioni quantistiche del vuoto. Inoltre, ha evidenziato che l'entropia non riguarda solo l'aumento della frazione di energia non più utile, ma anche la limitazione nell'estrazione di informazioni da un sistema fisico.

Negli ultimi decenni, il ruolo dell'energia è passato in secondo piano rispetto a quello dell'informazione, e il paradigma si sta spostando da quello energetico a quello che viene chiamato paradigma infocomputazionale. In questo nuovo paradigma, si tende a vedere l'universo come composto da supporti di informazione e da costruttori che operano su questi supporti. La ricerca teorica ha cercato di definire se ogni sistema fisico sia effettivamente un sistema computante o se esistano regole e requisiti tali per cui si possa considerare un sistema come computante.

Un risultato interessante di questa linea di ricerca è emerso grazie ai lavori di David Deutsch e Chiara Marletto, che hanno sviluppato la Teoria dei Costruttori (Constructor

Theory). Questa teoria propone un nuovo modo di formulare le leggi fondamentali della fisica in termini di compiti che possono o non possono essere realizzati, ovvero in termini di trasformazioni fisiche possibili o impossibili. Secondo la Teoria dei Costruttori, l'informazione è una proprietà fisica fondamentale che può essere definita oggettivamente e non ricorsivamente, senza la necessità di un osservatore o di un'intelligenza esterna.

Nei sistemi quantistici, l'informazione è intrinsecamente legata agli stati quantistici e alle loro trasformazioni. Questo contrasta con i sistemi classici, nei quali il concetto di informazione richiede necessariamente un osservatore per essere definito. Nel mondo quantistico, invece, l'informazione esiste indipendentemente dalla nostra capacità di osservarla o misurarla, ed è codificata nelle proprietà stesse dei sistemi fisici.

Questo nuovo sguardo sull'universo non può lasciare indifferenti e sembra fornire non solo una risposta al "cosa" ma anche al "perché" l'universo è ciò che è. L'informatica quantistica, quindi, non è solo la computazione eseguita da un computer quantistico, ma rappresenta l'informatica naturale dell'universo. Essa ci offre una nuova prospettiva per comprendere le leggi fondamentali della natura, basata sul concetto di informazione e sulle trasformazioni quantistiche possibili.

In questo contesto, l'informatica, la fisica e la matematica non sono più discipline separate, ma convergono verso una visione unificata in cui l'informazione gioca un ruolo centrale. I lavori di Deutsch e Marletto aprono la strada a una comprensione più profonda dei principi che governano il nostro universo, suggerendo che la realtà stessa potrebbe essere vista come un vasto processo computazionale.

L'informatica quantistica diventa così uno strumento fondamentale per esplorare e comprendere la struttura profonda della realtà. Non si tratta soltanto di sviluppare nuove tecnologie di calcolo, ma di abbracciare un paradigma che potrebbe rispondere a domande fondamentali sulla natura dell'esistenza e sulle leggi che regolano l'universo.

Nel cuore dell'era digitale, stiamo assistendo a una rivoluzione che ridefinirà il futuro: l'informatica quantistica. Non è semplicemente un passo avanti nella tecnologia; è un balzo quantico verso nuove possibilità. Questo libro nasce con l'obiettivo di guidarvi attraverso questa entusiasmante frontiera, fornendovi non solo la teoria essenziale ma anche un simulatore pratico per sperimentare immediatamente i concetti appresi.

## 1.2 Come utilizzare questo libro

Un aspetto importante di questo libro è l'introduzione al simulatore **QuantumSim**, trattato in dettaglio nell'ultimo capitolo. Questo simulatore è progettato per consentire ai lettori di applicare i concetti spiegati nei capitoli precedenti attraverso esercizi pratici.

### 1.2.1 Approccio Graduale

È possibile leggere il libro in due modalità principali:

- **Prima lettura**: Concentrarsi sulla comprensione dei concetti fondamentali senza eseguire gli esercizi. Questo approccio consente di costruire una solida base teorica e di familiarizzare con i principi dell'informatica quantistica.

- **Seconda lettura**: Una volta compresi i concetti di base e acquisita familiarità con il QuantumSim, rileggere il libro affrontando gli esercizi pratici proposti. Questo permetterà di consolidare le conoscenze teoriche attraverso l'applicazione pratica.

### 1.2.2 Navigazione tra i Capitoli

Il libro è strutturato in modo da fornire un equilibrio tra teoria e pratica. Mentre i primi capitoli introducono i fondamenti dell'informatica quantistica, gli ultimi capitoli si concentrano sull'applicazione di questi concetti mediante il QuantumSim. Si consiglia di leggere il libro seguendo l'ordine dei capitoli per garantire una progressione logica e graduale nell'apprendimento.

### 1.2.3 QuantumSim nell'ultimo capitolo

L'ultimo capitolo è interamente dedicato al QuantumSim, fornendo istruzioni dettagliate su come utilizzarlo per simulare circuiti quantistici. Questo capitolo rappresenta il ponte tra la teoria e la pratica, consentendo di sperimentare direttamente i concetti appresi. È un'opportunità unica per sviluppare competenze pratiche nel campo dell'informatica quantistica.

### Riferimenti bibliografici

La ricerca che ha portato all'elaborazione del paradigma infocomputazionale è ampiamente documentata. Uno dei testi più influenti in questo contesto è stato *The Fabric of Reality* di David Deutsch Deutsch [1997], dove viene introdotta la visione dell'universo come una serie di mondi paralleli regolati da leggi fisiche quantistiche. Questo si collega con i fondamenti dell'informatica quantistica che Deutsch ha esplorato nei suoi primi lavori Deutsch [1985]. Deutsch ha dimostrato che un computer quantistico può risolvere problemi che nessun computer classico potrebbe risolvere in modo efficiente, spingendo la ricerca verso una definizione più profonda dell'informazione.

Chiara Marletto ha ulteriormente sviluppato queste idee nel contesto della teoria dei costruttori. Nel suo lavoro Deutsch and Marletto [2015] e nel suo libro *The Science of Can and Can't* Marletto [2022], Marletto ha proposto che l'informazione non sia solo una proprietà astratta, ma una risorsa fondamentale che può essere trattata in modo oggettivo attraverso le leggi della fisica quantistica. La sua teoria amplia il paradigma infocomputazionale introducendo un quadro in cui l'informazione è centrale per comprendere i processi fisici stessi.

John Preskill Preskill [2018] ha contribuito all'idea che la computazione quantistica, anche nelle sue forme meno perfette, come quella attuale in cui si parla di NISQ (Noisy

Intermediate-Scale Quantum) devices, possa comunque rivoluzionare la nostra capacità di risolvere problemi complessi. La sua visione enfatizza il ruolo degli algoritmi quantistici nella futura evoluzione della tecnologia.

Le discussioni sulla decoerenza e sul passaggio dal mondo quantistico a quello classico trovano una solida base nel lavoro di Wojciech Zurek Zurek [2003], che ha esplorato come la perdita di coerenza quantistica porti alla percezione di un mondo classico, fenomeno cruciale per capire la transizione tra i due mondi.

Infine, Rolf Landauer Landauer [1991] ha gettato le basi per comprendere il legame indissolubile tra informazione e fisica, con la sua celebre affermazione che "l'informazione è fisica," sottolineando che ogni bit di informazione è legato a un processo fisico che richiede energia. Questo concetto è stato ulteriormente esplorato da Seth Lloyd Lloyd [2006], che ha proposto una visione dell'universo stesso come un gigantesco computer quantistico.

Per chi desidera esplorare in modo più profondo la filosofia dell'informazione e il suo impatto sulla scienza e sulla tecnologia, i lavori di Luciano Floridi Floridi [2011] offrono una visione dettagliata di come l'informazione sia diventata un concetto fondamentale in molte discipline, dalla fisica all'etica.

Queste fonti rappresentano solo un punto di partenza per esplorare il vasto campo della computazione quantistica e dell'informatica infocomputazionale, che continuano a evolversi man mano che le nuove scoperte vengono fatte nel campo della fisica quantistica e dell'informatica.

# Capitolo 2

# Dal Mondo Classico al Quantistico: Una Nuova Prospettiva

In questo capitolo, esploriamo il passaggio dalla fisica classica alla meccanica quantistica, delineando i principi fondamentali che differenziano queste due visioni del mondo fisico. Si parte dalla fisica classica, che si basa su concetti come il punto materiale e il determinismo, per poi passare alla fisica quantistica, che introduce elementi di indeterminazione e sovrapposizione degli stati. Viene sottolineato il concetto di misura, che in meccanica quantistica assume un ruolo centrale e modifica direttamente lo stato del sistema osservato.

Attraverso l'esperimento della doppia fenditura, si mette in evidenza la differenza fondamentale tra l'intuizione classica, che vede le particelle come oggetti dotati di traiettorie definite, e il comportamento quantistico, dove la sovrapposizione e l'interferenza giocano un ruolo chiave. L'esperimento illustra il principio di sovrapposizione, che è cruciale nell'informatica quantistica per permettere ai qubit di esistere in stati multipli simultaneamente.

Il capitolo si chiude affrontando l'impatto della meccanica quantistica nell'informatica, descrivendo come le sue caratteristiche uniche, come la sovrapposizione e l'entanglement, aprano nuove frontiere per la computazione rispetto a quella classica. Il simulatore QuantumSim viene presentato come strumento per esplorare questi concetti in modo interattivo, permettendo ai lettori di comprendere come la fisica quantistica possa essere applicata nell'informatica quantistica.

*"La scienza non è definitiva"*

La fisica è una scienza affascinante e impegnativa perché non si accontenta di risposte verosimili; esige una continua verifica sperimentale di ogni affermazione fatta in suo nome. Ironia della sorte, nonostante questo rigoroso approccio, ogni teoria è probabilmente destinata a essere superata, e quelle che crediamo vere sono in attesa di essere dimostrate false o, per meglio dire, *falsificate*. Dobbiamo quindi rinunciare allo studio delle teorie fisiche? Assolutamente no, perché entro certi limiti queste teorie funzionano e ci permettono di comprendere il mondo che ci circonda.

Nel XX secolo, ad esempio, è stato dimostrato che la teoria della gravitazione universale di Newton non è completamente corretta; tuttavia, rimane estremamente utile per numerosi calcoli di balistica spaziale e per le approssimazioni più *grossolane* nell'astronomia del sistema solare. La teoria di Newton è stata sostituita dalla relatività generale di Einstein, ma anche quest'ultima, negli ultimi decenni, ha mostrato segni di cedimento e in futuro potrebbe essere messa in discussione.

Le leggi che la fisica identifica come fondamentali costituiscono la *meccanica* del nostro universo. Il primo incontro *scolastico* con la meccanica avviene attraverso la *cinematica*, che descrive le relazioni tra posizione e velocità di un *punto materiale*. Il concetto di punto materiale è un concetto *classico* basato sull'idea che la materia possa essere pensata in modo analogo alla geometria euclidea, dove i solidi hanno superficie e volume definiti, ma sono costituiti da insiemi di punti ognuno dei quali ha dimensione nulla.

Allo stesso modo, il punto materiale ha dimensione nulla, ma ad esso è associata una massa. Lo stato del punto materiale è completamente descritto quando sono definite le sue tre coordinate spaziali $x$, $y$ e $z$, e le tre componenti cartesiane della sua velocità: $v_x$, $v_y$ e $v_z$. La posizione e la velocità del punto materiale possono variare con il tempo e, per questo, si specifica che sia le coordinate che le velocità sono funzioni del tempo: $x(t)$, $y(t)$, $z(t)$ e $v_x(t)$, $v_y(t)$, $v_z(t)$.

La meccanica quantistica trova il suo primo **disaccordo** con la meccanica classica già a questo livello: secondo la meccanica quantistica, non è possibile che, in un dato istante di tempo, per una particella si conoscano **esattamente** sia la sua posizione che la sua velocità. Questo è noto come *principio di indeterminazione di Heisenberg*.

Come abbiamo chiarito all'inizio del paragrafo, la meccanica quantistica si discosta notevolmente dalla meccanica classica, ma non può essere definita senza di essa. Nella meccanica classica, un sistema fisico si ritiene completamente definito quando sono note tutte le posizioni dei punti che lo compongono, tutte le velocità e le eventuali forze che stanno agendo. Ovviamente, questo non può essere lo stesso per un sistema quantistico, poiché queste grandezze non possono essere note simultaneamente con precisione arbitraria.

La meccanica quantistica introduce all'interno della teoria anche il concetto di **misura**. È importante notare che anche la teoria classica prevedeva le misure sperimentali; anzi, queste fanno parte del metodo sperimentale con cui la teoria è stata costruita. Tuttavia, nella meccanica classica, tali misure sono intese come esterne alla teoria.

La differenza tra i due approcci è che, nel *classico*, il processo di misura è visto come un'azione necessaria all'osservatore dell'esperimento per conoscere empiricamente il valore di una grandezza che sta osservando, ma tale grandezza ha un ben preciso valore indipendente dal processo di misura. Nel *quantistico*, invece, secondo l'interpretazione della scuola di Copenaghen, le grandezze fisiche di un dato sistema non hanno sempre valori definiti, ma li assumono nel momento stesso in cui avviene un evento *speciale*, come appunto una misura.

A questo proposito, è d'obbligo chiarire subito per evitare malintesi. Il concetto di **misura** in meccanica quantistica non prevede la presenza di un essere intelligente: per

misura si intende, ad esempio, l'interazione di un sistema molto piccolo, cioè di dimensioni atomiche, subatomiche o particellari, con un sistema classico, cioè di grande massa. Non si pensi che "grande massa" significhi la massa di un edificio; basta prendere venti grammi di acqua per avere (circa) un numero di Avogadro $\left(6 \times 10^{23}\right)$ di molecole, quindi il concetto di grande massa va rapportato alla scala atomica (Landau, 1947; Bohr, 1928).

Questo aspetto della teoria è difficile da comprendere perché si scontra con l'esperienza quotidiana, che è appunto un'esperienza *classica* della fisica.

## 2.1 Dal Mondo Classico al Quantistico: L'Esperimento della Doppia Fenditura

Se lasciamo cadere un oggetto dalla mano, siamo convinti che, durante la sua caduta, esso abbia una posizione e una velocità ben definite in ogni istante. Non importa se lo stiamo osservando o meno: secondo il senso comune, l'oggetto si trova sempre in uno stato fisico determinato.

Ma cosa accade nel mondo quantistico? Un esperimento fondamentale che mette in luce le differenze tra la fisica classica e quella quantistica è l'**esperimento della doppia fenditura**.

### 2.1.1 L'Esperimento della Doppia Fenditura

Immaginiamo di avere una sorgente che emette particelle, come elettroni o fotoni, dirette verso una barriera con due fenditure molto sottili e vicine tra loro. Dietro la barriera, posizioniamo uno schermo che rileva l'arrivo delle particelle.

Secondo l'intuizione classica, ci aspetteremmo che le particelle passino attraverso una delle due fenditure, formando sullo schermo due accumuli distinti, corrispondenti alle posizioni delle fenditure.

Tuttavia, quando eseguiamo l'esperimento senza osservare quale fenditura viene attraversata dalle particelle, sullo schermo appare una **figura di interferenza**, caratteristica tipica delle onde. Questa figura consiste in una serie di frange alternate di massimi e minimi, indicando che le particelle si comportano come onde che interferiscono tra loro.

Ma se installiamo un dispositivo per rilevare attraverso quale fenditura passa ogni particella, l'interferenza scompare. Sullo schermo, osserviamo due bande distinte, come previsto dalla fisica classica. L'atto di misurare la fenditura attraversata modifica il comportamento delle particelle.

### 2.1.2 Il Principio di Sovrapposizione degli Stati

Questo fenomeno illustra il **principio di sovrapposizione degli stati** in meccanica quantistica. Prima della misurazione, le particelle non hanno una traiettoria definita attraverso una singola fenditura; invece, esistono in una sovrapposizione di stati in cui passano attra-

verso entrambe le fenditure simultaneamente. Solo quando si effettua una misurazione, la sovrapposizione collassa in uno stato definito.

### 2.1.3 Implicazioni per l'Informatica Quantistica

Il principio di sovrapposizione è fondamentale nell'informatica quantistica, poiché consente ai qubits di esistere in più stati contemporaneamente, aumentando esponenzialmente la capacità di calcolo rispetto ai bits classici. Questo fenomeno è alla base di algoritmi quantistici che promettono di risolvere problemi complessi in modo più efficiente.

Approfondiremo questi concetti nei capitoli successivi, esplorando come la meccanica quantistica rivoluziona il modo in cui elaboriamo e trasmettiamo le informazioni.

## 2.2 Meccanica e computazione quantistica

Per comprendere a fondo la meccanica quantistica servono almeno tre anni di studio dedicato: uno per le basi matematiche, partendo dal calcolo differenziale in una sola dimensione sul campo dei numeri reali fino al calcolo sul campo dei numeri complessi; uno per lo studio della meccanica hamiltoniana; e uno per la meccanica quantistica, sia nella teoria non relativistica che relativistica. Questo percorso **non può** essere affrontato come passatempo serale; richiede un impegno a tempo pieno.

Fortunatamente, è possibile comprendere i principi dell'informatica quantistica e la base del funzionamento di un computer quantistico anche senza conoscere tutta la meccanica quantistica. La situazione attuale non è diversa dallo scenario che si presentò con l'avvento dei primi computer nel secolo scorso.

Le prime macchine erano la sintesi perfetta dello stato dell'arte della logica matematica e dell'elettronica, discipline allora riservate quasi esclusivamente ai fisici, poiché i corsi di ingegneria elettronica ancora non esistevano. Negli anni '50 del XX secolo, per spiegare il funzionamento di un computer era necessario un fisico esperto, e per scrivere un programma serviva comunque un matematico o un ingegnere. Il concetto di memoria volatile era assolutamente nuovo e si basava su tecnologie al limite del realizzabile, rendendo difficile separare l'idea astratta di **automa** dalla sua concreta realizzazione. Come risultato, la programmazione di un computer appariva possibile solo ad esperti fisici, matematici ed ingegneri.

Anche senza andare troppo indietro nel tempo, basta pensare al film "War Games" di John Badham (1983). Il film racconta la storia di un giovane hacker che si trova ad affrontare un'intelligenza artificiale che scambia la guerra mondiale per un gioco di strategia. È interessante notare come il programma di intelligenza artificiale venga presentato come il frutto del lavoro di un unico scienziato: Stephen Falken. L'idea dello *scienziato* capace da solo di realizzare *un mostro* come quello del dottor Frankenstein era un retaggio dell'informatica dei primi anni '50.

Dalla percezione di un'informatica distante ed *aliena*, in pochi decenni siamo passati a linguaggi di programmazione come *Scratch*, pensati per introdurre la programmazione già dalle scuole elementari. Oggi è possibile formare un programmatore in pochi mesi senza che egli conosca nulla di elettronica. Per programmare un computer classico è necessario prendere confidenza con la tecnologia—tastiera, monitor, editor, ecc.—e apprendere il concetto di *linguaggio formale* con il quale è possibile descrivere un algoritmo, anche senza sapere come funziona effettivamente un computer.

Quanto appena detto per la programmazione di computer classici vale anche per i computer quantistici. Se si conosce la meccanica quantistica, si può comprendere a fondo come funzioni la loro tecnologia, ma una comprensione approfondita non è necessaria per formulare un algoritmo quantistico. Il messaggio è il seguente: con un certo sforzo, chiunque si applichi può comprendere e applicare la computazione quantistica e, se deciderà di studiare in modo completo la meccanica quantistica, comprenderà anche come funziona un computer quantistico. Non è molto diverso dalla situazione attuale.

In questo testo svilupperemo diversi esempi di programmi che simuleranno alcuni aspetti di un computer quantistico. Va da sé che la simulazione non potrà riprodurre le prestazioni in termini di velocità. A tal fine, introdurremo **QuantumSim**, un simulatore progettato per aiutare a visualizzare e comprendere i concetti fondamentali dell'informatica quantistica. QuantumSim ci permetterà di esplorare questi concetti in modo interattivo e intuitivo, facilitando il processo di apprendimento.

**Domanda n. 2** Quali sono i tre aspetti peculiari dell'informatica quantistica di cui si è accennato?

1. Dense coding, quantum teleportation, indeterminazione degli stati
2. Dense coding, quantum teleportation, no cloning principle
3. Dense coding, no cloning principle, sovrapposizione degli stati

## 2.3 Simulare la fisica sul computer: si può fare, Mr. Feynman?

Il titolo di questa sezione nasce dalla parafrasi del libro "Sta scherzando, Mr. Feynman?" di Ralph Leighton, amico di Richard Feynman, che raccolse diversi aneddoti sul celebre scienziato. Richard Feynman era un fisico con una ricca carriera accademica, che aveva lavorato con John von Neumann a metà degli anni '40 e fu testimone della nascita dei primi computer *classici*.

### 2.3.1 L'Idea Rivoluzionaria di Feynman

L'idea di un modello di computazione basato sulla meccanica quantistica risale almeno al 1980, proprio grazie a Feynman. Nel 1981 presentò la sua idea con un articolo intitolato

*"Simulating Physics with Computers"*, pubblicato sull'*International Journal of Theoretical Physics*, nel quale poneva una domanda precisa:

*È possibile simulare la fisica quantistica usando un computer?*

Nell'articolo, Feynman spiegava le ragioni per cui i computer classici erano adatti a simulare la fisica classica, ma non la fisica quantistica. Questa riflessione aprì la strada all'idea di sviluppare computer basati sui principi della meccanica quantistica.

### 2.3.2 L'Influenza di von Neumann e gli Automi Cellulari

È interessante notare l'influenza che John von Neumann ebbe sull'allora giovane Feynman. Nell'articolo, infatti, Feynman si riferisce ai computer in termini di *automi cellulari*, un'idea originata proprio da von Neumann mentre cercava di progettare un modello di automa capace di *auto-riprodursi*.

Gli automi cellulari sono sistemi computazionali in cui il valore di ogni cella (o bit) evolve in funzione dei valori delle celle vicine, secondo regole locali e deterministiche. Questa concezione del calcolo enfatizza il principio di **località**, fondamentale nella fisica classica.

### 2.3.3 Verso la Meccanica Quantistica e oltre

L'argomentazione usata da Feynman è importante perché ci introduce direttamente ai principi della meccanica quantistica necessari per sviluppare il resto del testo, in particolare al concetto di **non-località** che caratterizza la fisica quantistica rispetto alla fisica classica.

Grazie alle idee pionieristiche di Feynman, oggi possiamo esplorare la fisica quantistica attraverso strumenti di simulazione come **QuantumSim**. Questi strumenti ci permettono di superare le limitazioni dei computer classici nella simulazione di sistemi quantistici, avvicinandoci alla visione di Feynman di un calcolatore quantistico.

### 2.3.4 La fisica classica è locale

La fisica classica è una *teoria locale*, cioè l'interazione tra due *entità fisiche* avviene solo quando sono a diretto contatto. Questa affermazione potrebbe sembrare in contrasto con i fenomeni elettromagnetici ed elettrostatici: pensiamo, per esempio, a due punti materiali che esercitano l'uno sull'altro una forza coulombiana. In realtà, anche l'elettromagnetismo è una teoria locale. L'azione a distanza tra due cariche è dovuta al *campo di forze* generato da ciascuna carica, che agisce localmente sull'altra. L'azione su ogni carica dipende solo dal valore del campo nel punto in cui essa si trova, rendendo l'interazione effettivamente **locale**.

Per chiarire questo concetto, Feynman fece riferimento al modello computazionale dell'**automa cellulare**. A differenza della più nota *architettura di von Neumann*, l'automa cellulare ha la caratteristica che ogni bit cambia il proprio valore in base al valore

dei bit adiacenti, rendendo l'*interazione locale* evidente. Anche i computer odierni, basati sull'architettura di von Neumann, sono in realtà *locali*, ma nell'automa cellulare questa località è resa più esplicita, motivo per cui Feynman preferì riferirsi ad esso.

### Il modello di calcolo dell'automa cellulare

Questo modello fu introdotto da von Neumann, che, studiando i primi modelli di intelligenza artificiale nel campo informatico, ideò l'automa cellulare per realizzare una forma di automa capace di auto-riprodursi. La particolarità dell'automa cellulare è che la computazione avviene in modo *parallelo* e puntuale, o, appunto, **locale**. L'automa consiste in:

- Una stringa di bit che possono essere nello stato 0 oppure 1, dove ogni bit è chiamato *cella*.
- Una regola che stabilisce come ogni cella cambi stato in base al valore delle celle vicine.

Il sistema di calcolo è molto semplice. All'istante $t = 0$, l'automa si trova in una certa configurazione iniziale. All'istante $t = 1$, per ogni cella, si valutano i bit adiacenti (quello a destra e quello a sinistra) e, in base alla regola stabilita, si assegna il nuovo valore alla cella per la configurazione a $t = 1$. La procedura si ripete per i tempi successivi.

Per fare un esempio, consideriamo la seguente configurazione:

- **Configurazione iniziale**: 1 1 0 1 1 0 1 1
- **Regola**:
  - 1 **1** 1 → **0**
  - 1 **1** 0 → **1**
  - 1 **0** 1 → **1**
  - 1 **0** 0 → **0**
  - 0 **1** 1 → **1**
  - 0 **1** 0 → **1**
  - 0 **0** 1 → **1**
  - 0 **0** 0 → **0**

La computazione dei primi quattro passi è rappresentata nella tabella seguente:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $t = 0$ | (1) | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | (1) |
| $t = 1$ | (0) | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | (0) |
| $t = 2$ | (0) | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | (1) |
| $t = 3$ | (1) | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | (1) |

I bit fra parentesi sono usati per il *padding*, cioè per gestire le celle ai bordi. In questo esempio, il padding è ottenuto per *circolarità*: è come se la stringa di bit si chiudesse su sé stessa. Cambiando le regole, è possibile definire diversi tipi di automi, anche su una griglia bidimensionale. Quello mostrato qui è noto come **Rule 110**: un automa Turing completo che, in linea di principio, può essere usato per eseguire qualsiasi programma che può essere eseguito su un calcolatore ordinario.

Tornando al punto, Feynman illustrò come, sfruttando la *località* del computer, sia possibile **simulare** la fisica classica, ma, al contrario, **non** sia possibile simulare la fisica regolata dalla meccanica quantistica.

È importante chiarire che, su un computer classico, può essere eseguita l'evoluzione di un sistema quantistico: se si conosce la dinamica di un sistema quantistico, cioè le equazioni che descrivono la sua evoluzione nel tempo (l'analogo delle leggi orarie delle coordinate nella fisica classica), è assolutamente possibile **emularlo**, ma non **simularlo** nel senso pieno del termine. Questo perché la meccanica quantistica presenta fenomeni di **non-località**. Questo argomento è complesso e può dar luogo a fraintendimenti. La non-località della meccanica quantistica non implica che sia ammessa un'azione a distanza nel senso classico, ma che esistano correlazioni tra misure sperimentali che non sono spiegabili da nessuna teoria locale. La non-località è legata al concetto di *entanglement*, che verrà discusso nei prossimi capitoli.

Comprendere questo argomento è fondamentale, poiché è stata una delle principali motivazioni che ha spinto Feynman e altri, insieme e dopo di lui, a indagare la possibilità di realizzare una macchina quantistica che permettesse di simulare la fisica quantistica, così come i computer classici permettono di simulare la fisica classica.

Grazie a questi sforzi, oggi abbiamo strumenti come **QuantumSim**, che ci aiutano a esplorare e comprendere i fenomeni quantistici, superando le limitazioni dei computer classici nella simulazione dei sistemi non-locali.

## 2.4 Bibliografia ragionata

In questo capitolo, introduciamo il passaggio cruciale dalla meccanica classica alla meccanica quantistica, un cambiamento che ha rimodellato la nostra comprensione della realtà fisica e ha fornito le basi per l'informatica quantistica. Per approfondire questi concetti, alcuni testi fondamentali risultano indispensabili.

Il lavoro di Landau and Lifshitz [1958], *Quantum Mechanics: Non-Relativistic Theory*, è essenziale per una solida comprensione della meccanica quantistica di base. Questo testo classico introduce formalmente i principi della meccanica quantistica e il concetto di sovrapposizione, che è fondamentale per il funzionamento dei qubits. È il punto di partenza per chiunque voglia padroneggiare i fondamenti matematici e fisici necessari per l'informatica quantistica.

Le implicazioni del principio di indeterminazione e del ruolo della misura quantistica, concetti discussi nel capitolo, trovano una trattazione approfondita nel lavoro di Bohr [1928], *The Quantum Postulate and the Recent Development of Atomic Theory.* Bohr, uno degli architetti della meccanica quantistica, chiarisce come l'interazione tra sistemi quantistici e classici giochi un ruolo centrale nel determinare gli esiti osservabili, concetto che si applica direttamente anche alla computazione quantistica, dove la misura influenza il risultato dell'elaborazione.

Per comprendere come la fisica quantistica possa essere simulata utilizzando il calcolo quantistico, il lavoro di Feynman [1982], *Simulating Physics with Computers*, è essenziale. Feynman è stato il primo a suggerire l'idea che la fisica quantistica potesse essere simulata solo da un computer quantistico, aprendo la strada all'idea stessa di costruire calcolatori che utilizzano principi quantistici. Questo concetto è alla base dello sviluppo di strumenti come QuantumSim, che permettono di simulare l'informatica quantistica.

L'idea rivoluzionaria di un calcolatore quantistico universale trova fondamento nel lavoro di Deutsch [1985], *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer.* Deutsch sviluppa l'idea che i computer quantistici possano risolvere problemi irrisolvibili dai computer classici e pone le basi teoriche per comprendere come i principi della meccanica quantistica possano essere applicati alla computazione.

Infine, il lavoro di Marletto [2022], *The Science of Can and Can't: A Physicist's Journey Through the Land of Counterfactuals*, offre una prospettiva unica sul ruolo della computazione nell'universo. Marletto esplora la nozione di "informazione oggettiva" nei sistemi quantistici, sottolineando come il paradigma infocomputazionale possa riformulare i principi della fisica. La sua ricerca amplia il dibattito su come i concetti di computazione e informazione possano essere applicati non solo all'informatica quantistica, ma anche alla fisica fondamentale.

Per una panoramica più accessibile e moderna, Aaronson [2013], *Quantum Computing Since Democritus*, è una risorsa chiave per collegare i principi della logica classica e della teoria quantistica, rendendo il passaggio tra i due mondi più comprensibile a un pubblico più vasto, e facilitando la comprensione delle basi dell'informatica quantistica.

Questi lavori formano la struttura teorica e pratica su cui si fonda il capitolo e sono fonti imprescindibili per chi desidera approfondire la transizione dal mondo classico al quantistico e il suo impatto sul calcolo.

### 2.4.1 Esercizio pratico con QuantumSim

Usiamo **QuantumSim** per esplorare il principio di sovrapposizione. Scriviamo un programma che simula lo stato di un qubit prima e dopo l'applicazione di un gate Hadamard, che genera una sovrapposizione degli stati:

```c
#include <stdio.h>
#include "../src/quantum_sim.h"

void circuit(void){
    QubitState* state = initializeState(1); // Stato iniziale con un qubit
    printState(state);                      // Stampa lo stato iniziale |0>
    applyHadamard(state, 0);               // Applica il gate Hadamard
    printState(state);                      // Stampa lo stato in
    sovrapposizione
    measure(state, 0);                     // Misura il qubit
    printState(state);                      // Stampa lo stato post-misura
    freeState(state);                       // Libera la memoria
    return 0;
}
```

Listing 2.1: circuit2.c

**Spiegazione:**

- Lo stato iniziale del qubit è $|0\rangle$.
- Dopo l'applicazione del gate Hadamard, il qubit è in una sovrapposizione: $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.
- La misura collassa lo stato in $|0\rangle$ o $|1\rangle$, con probabilità $\frac{1}{2}$ ciascuna.

Compilate ed eseguite il programma per osservare come il principio di sovrapposizione si traduce in pratica e come la misura influenza lo stato quantistico.

## Transizione verso il Capitolo 3

Ora che abbiamo introdotto il principio di sovrapposizione e il suo impatto sull'informatica quantistica, nel **Capitolo 3** esploreremo i fondamenti matematici della meccanica quantistica. Questi strumenti ci permetteranno di formalizzare ulteriormente i concetti trattati finora e prepararci a implementare algoritmi quantistici complessi.

# Capitolo 3

# Dai bits classici ai qubits quantistici

## 3.1 Introduzione

Il capitolo "Dai bits classici ai qubits quantistici" introduce uno degli argomenti centrali della computazione quantistica: il passaggio dai bit classici, utilizzati nei computer tradizionali, ai qubit, che sfruttano le leggi della meccanica quantistica. Questo capitolo fornisce una panoramica chiara delle differenze tra informazione classica e quantistica, spiegando concetti chiave come la sovrapposizione degli stati, l'entanglement e la non-località, che permettono alla computazione quantistica di risolvere problemi in modo esponenzialmente più efficiente rispetto a quanto è possibile con i computer tradizionali.

La transizione dal mondo classico a quello quantistico segna un cambio radicale di paradigma: mentre un bit classico può assumere solo i valori 0 o 1, un qubit può esistere in una sovrapposizione di stati, consentendo una rappresentazione dell'informazione molto più ricca. Questo capitolo è cruciale per comprendere le basi teoriche su cui poggia la computazione quantistica e prepara il lettore agli argomenti più avanzati che saranno trattati in seguito.

Per esplorare questi concetti, si parte da una descrizione dei bit nell'informatica classica e si arriva a un'analisi dettagliata dei qubits e dei sistemi fisici che li rappresentano, come i fotoni. Viene anche presentato il concetto di misura nel contesto quantistico, con una particolare attenzione al principio di indeterminazione e alla natura probabilistica delle osservazioni quantistiche.

Nel corso di questo capitolo, esploreremo:

- La transizione dai **bits classici** ai **qubits quantistici**, evidenziando le differenze fondamentali tra l'informazione classica e quella quantistica. - Il concetto di **sovrapposizione degli stati** e come esso rivoluziona la nostra comprensione dell'informazione. - Introduzione a principi quantistici chiave come l'**entanglement** e la **non-località**. - Come questi principi possono essere applicati nella computazione quantistica per sviluppare algoritmi più potenti.

Strumenti come **QuantumSim** ci aiuteranno a visualizzare e comprendere questi concetti, rendendo più accessibile l'apprendimento della computazione quantistica.

Preparatevi quindi a immergervi nel mondo affascinante dei qubits, dove le leggi della fisica quantistica aprono nuove possibilità per l'elaborazione dell'informazione.

## 3.2 I bits nell'informatica classica

Un **bit** è l'unità fondamentale di informazione utilizzata nell'informatica classica. Un bit può assumere solo due valori: 0 oppure 1. Utilizzando più bit, è possibile rappresentare informazioni complesse. Ad esempio, il numero decimale 11 può essere rappresentato in forma binaria come la sequenza di bit 1011. Questa sequenza corrisponde al calcolo:

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11.$$

Un bit può essere *realizzato* con diverse tecnologie. In generale, si tratta di creare un **sistema fisico** che ammetta due possibili **stati** di equilibrio. Chiameremo questo sistema **sistema fisico classico**, dove l'aggettivo "classico" si riferisce alla possibilità di descrivere la fisica del sistema usando le leggi della meccanica classica (per intenderci, quelle di Newton).

Ad esempio, possiamo pensare a un bit realizzato con una matita, in cui il valore 1 è associato allo **stato** in cui la matita è in posizione verticale, e il valore 0 allo **stato** in cui la matita è in posizione orizzontale.

Lo stato *verticale* della matita può essere espresso matematicamente indicando con $\theta$ l'angolo tra la matita e il piano (ad esempio, il piano del tavolo). Se $\theta$ è uguale a $\frac{\pi}{2}$, allora il bit è 1; se $\theta$ è zero, allora il bit è 0. In pratica, associamo il valore *binario* del bit allo stato $\theta$ del sistema fisico, che in questo caso è la nostra semplice matita.

Esistono molti altri modi per realizzare un bit. Ad esempio, utilizzando un interruttore collegato a una luce LED, possiamo associare al LED acceso il valore 1 e al LED spento il valore 0. Ovviamente, per condensare molti gigabit in uno spazio limitato, non possiamo usare matite o LED, ma utilizziamo tecnologie più avanzate, come quelle presenti nei moderni calcolatori.

### 3.2.1 Proprietà dei bits

In questo paragrafo, ci soffermiamo su alcune osservazioni che possono sembrare ovvie, ma che acquisiranno interesse quando le confronteremo con osservazioni analoghe sui **qubits**, i bit quantistici. Per semplicità, continuiamo a considerare i bit identificati dallo stato $\theta$ della matita, ma le considerazioni che trarremo valgono in generale per qualsiasi tecnologia *classica* utilizzata per realizzarli.

### 3.2.2 Relazione tra lo stato di un bit e il suo valore

Lo stato di un *bit-matita* è sempre perfettamente determinato. La matita può trovarsi solo nello stato $\theta = \frac{\pi}{2}$ (verticale) o nello stato $\theta = 0$ (orizzontale) e, a causa della forza di

gravità, non è possibile che si trovi in equilibrio in uno stato intermedio. Per conoscere il valore del bit, è sufficiente misurare il suo angolo rispetto al piano.

Se un osservatore sulla Terra si trovasse ruotato rispetto al piano di un angolo $\alpha$, vedrebbe il *bit-matita* formare un angolo diverso dai due angoli possibili appena definiti. Dal punto di vista dell'osservatore, infatti, la matita si troverebbe agli angoli $\theta = \frac{\pi}{2} + \alpha$ o $\theta = 0 + \alpha$.

Questa rotazione non impedirebbe di determinare il corretto valore del bit. L'osservatore saprebbe di essere ruotato perché potrebbe verificarlo con un esperimento legato alla forza gravitazionale, ad esempio osservando la direzione in cui gli oggetti cadono a terra. In tal modo, potrebbe sempre ricondurre la propria misurazione dell'angolo della matita nel suo sistema di riferimento all'angolo $\theta$ utilizzato per definire lo stato della matita.

Quindi, per i bit *classici*, possiamo affermare che:

*Lo stato di un bit è associato sempre a uno e un solo valore: 0 oppure 1.*

### 3.2.3 Modifica dello stato di un bit

Lo stato di un *bit-matita* può essere modificato solo in seguito a un'azione volontaria che lo porti dallo stato $\theta = 0$ a $\theta = \frac{\pi}{2}$ oppure da $\theta = \frac{\pi}{2}$ a $\theta = 0$.

### 3.2.4 Copia dello stato di un bit

Lo stato di un bit è sempre noto ed è possibile eseguirne una copia senza modificare l'originale. Questa caratteristica permette molte operazioni utili, come la copia di un documento da condividere o da inviare per email, e altre operazioni delicate dal punto di vista della sicurezza informatica. Ad esempio, un intruso potrebbe leggere un messaggio scambiato tra due soggetti comunicanti senza che loro se ne accorgano, intercettando la comunicazione e copiandone il contenuto bit a bit, senza compromettere il messaggio trasmesso.

### 3.2.5 Stato di più bits

Il bit è l'unità minima di informazione dell'informatica discreta. Per rappresentare grandezze complesse, come ad esempio per codificare un alfabeto o i numeri decimali, un singolo bit non è sufficiente. È quindi necessario utilizzare un insieme di bit, in cui ogni bit, a seconda della sua posizione, ha un significato preciso, come visto nell'esempio della trasformazione della sequenza binaria 1011 nel numero decimale 11.

Consideriamo, ad esempio, un insieme di otto *bit-matita* (un byte), disposti alcuni nello stato $\theta = 0$ e altri nello stato $\theta = \frac{\pi}{2}$. Per riferirci in modo chiaro allo stato di ognuna delle matite, useremo l'indice $i$. Ad esempio, scrivendo $\theta_0$ indicheremo lo stato della prima matita, mentre scrivendo $\theta_7$ quello dell'ottava.

Usando queste otto matite, creiamo un nuovo sistema fisico che ammette $2^8$ configurazioni diverse, ognuna delle quali corrisponde a una precisa sequenza dei valori $\theta_i$. Ad

esempio, lo stato:

$$(\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7) = \left(\frac{\pi}{2}, 0, 0, 0, 0, 0, 0, \frac{\pi}{2}\right)$$

corrisponde ai bits:

$$(b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7) = (1, 0, 0, 0, 0, 0, 0, 1)\,.$$

La corrispondenza individuata è biunivoca, e quindi alle 256 diverse configurazioni corrispondono esattamente 256 diverse informazioni codificate dagli otto bits, non di più e non di meno.

Abbiamo portato volutamente l'attenzione sulla relazione tra l'unità di informazione (0 e 1) e il concetto di sistema fisico classico, perché nel prossimo paragrafo riprenderemo questa relazione, applicandola però al concetto di sistema fisico **quantistico**.

## 3.3 I qubits

### 3.3.1 Sistemi fisici classici e quantistici

Ci chiediamo ora se esiste una differenza tra sistemi fisici *classici* e sistemi *quantistici*. A rigor di logica, la teoria quantistica è considerata la meccanica che regola ogni sistema fisico esistente; quindi, in senso stretto, esistono solo sistemi quantistici. Tuttavia, è prassi indicare come quantistici solo quei sistemi che non possono essere spiegati con sufficiente accuratezza dalla meccanica classica.

Ad esempio, per descrivere il moto di una palla di cannone lanciata con un certo angolo di inclinazione, non abbiamo bisogno della meccanica quantistica; un tale sistema viene definito *classico*. Chiamiamo quindi sistemi quantistici quei sistemi fisici che richiedono la meccanica quantistica per essere descritti correttamente.

Sistemi come gli atomi e le particelle elementari hanno sempre natura quantistica. Anche certi sistemi *macroscopici*, normalmente non presenti in natura sulla Terra ma ottenuti artificialmente, possono essere considerati quantistici se il loro comportamento non è descrivibile dalla fisica classica e richiede l'uso della meccanica quantistica.

### 3.3.2 La sfida nella misurazione dei sistemi quantistici

Realizzare un esperimento per misurare un sistema quantistico non è semplice come farlo per un sistema classico. Gli esperimenti per acquisire misure classiche, come la posizione di un oggetto o il tempo di percorrenza di un carrello lungo un binario, sono facilmente realizzabili e fanno parte del bagaglio di studio ed esperienza che si acquisisce già a scuola. Per questo, il concetto di **stato** introdotto con l'esempio della matita è diretto e intuitivo, rientrando nell'esperienza comune di tutti.

Al contrario, il concetto di **stato** per i sistemi fisici quantistici non è altrettanto intuitivo, poiché nella vita quotidiana mancano le occasioni per familiarizzare con esperimenti di

misura quantistica. I fenomeni quantistici si manifestano su scale atomiche e subatomiche, al di fuori della nostra esperienza diretta.

### 3.3.3 Dal bit classico al qubit quantistico

Un **bit** è un'associazione tra uno stato fisico classico e uno dei due valori numerici 0 e 1. Analogamente, un **qubit** (quantum bit) è un'associazione tra uno stato fisico quantistico e i valori 0 e 1. La differenza fondamentale tra lo stato fisico classico e quello quantistico è ciò che determina la distinzione tra informatica classica e quantistica.

Per definire i qubits, nel seguito di questo capitolo introdurremo le particelle chiamate **fotoni**, utilizzando la misura del loro stato fisico per associarvi il valore binario 0 o 1. I fotoni sono esempi di sistemi quantistici che possiamo sfruttare per rappresentare qubits.

Grazie a strumenti come **QuantumSim**, possiamo esplorare e visualizzare il comportamento dei qubits, aiutandoci a comprendere concetti che non sono immediatamente intuitivi.

## 3.4 I fotoni: le particelle del campo elettromagnetico

### 3.4.1 Introduzione ai fotoni

I **fotoni** sono le particelle elementari che costituiscono il *quanto* del campo elettromagnetico. Essi sono stati introdotti nella fisica teorica attraverso un procedimento noto come *seconda quantizzazione*.

### 3.4.2 Prima e seconda quantizzazione

La **prima quantizzazione** è il procedimento con cui si ottiene la descrizione quantistica di un sistema fisico a partire dalla sua descrizione classica. Questo approccio è applicato per descrivere particelle come l'elettrone o il positrone, utilizzando equazioni come quella di Schrödinger.

La **seconda quantizzazione**, invece, si applica ai campi fisici, come il campo elettromagnetico. Attraverso questo procedimento, i campi classici vengono quantizzati, portando all'introduzione di particelle quantistiche associate ai campi stessi. Nel caso del campo elettromagnetico, la seconda quantizzazione porta all'emergere dei fotoni come quanti del campo.

### 3.4.3 La natura dei fotoni

Ogni *onda piana* del campo elettromagnetico può essere vista come un insieme di fotoni, particelle prive di massa ma con energia, quantità di moto e polarizzazione definite. Già all'inizio del XX secolo, la fisica sperimentale aveva evidenziato la natura *quantizzata* dello scambio di energia tra atomi e campo elettromagnetico. Albert Einstein ricevette il premio

Nobel per aver spiegato l'*effetto fotoelettrico*, descrivendo come il campo elettromagnetico possa estrarre elettroni dagli atomi cedendo energia in quantità discrete, chiamate fotoni.

Per questo motivo, i fotoni, inizialmente introdotti come risultato matematico dalla quantizzazione del campo elettromagnetico, hanno rapidamente acquisito lo status di particelle *reali*, al pari di elettroni e protoni.

### 3.4.4 Come immaginare un fotone?

Ma *come possiamo immaginare un fotone se non ha massa?* La meccanica quantistica rinuncia a una descrizione visiva o intuitiva delle particelle e dei campi. Si concentra invece sulla previsione dei risultati degli esperimenti. Un fotone è definito dalle sue proprietà misurabili, come energia, impulso e polarizzazione, senza bisogno di una rappresentazione pittorica.

Anche se potremmo essere tentati di immaginare i fotoni come piccole particelle che trasportano energia viaggiando alla velocità della luce, questo può portare a fraintendimenti. Ai fini della comprensione dell'informatica quantistica, non è necessario visualizzare il fotone in termini classici; anzi, ciò potrebbe risultare controproducente. La meccanica quantistica non è una teoria intuitiva, e i tentativi di renderla tale spesso generano confusione.

### 3.4.5 La polarizzazione della luce e dei fotoni

Per comprendere lo **stato** di un fotone, è utile partire dalle onde elettromagnetiche nella teoria classica. Le onde elettromagnetiche sono oscillazioni dei campi elettrico e magnetico che si propagano nello spazio. Un esempio comune è la luce visibile, che permette all'occhio umano di percepire l'ambiente circostante.

Una proprietà fondamentale delle onde elettromagnetiche è la **polarizzazione**, che descrive la direzione di oscillazione del campo elettrico. Nel caso di un'onda piana e polarizzata linearmente, questa direzione rimane costante nel tempo.

Esempi di fenomeni legati alla polarizzazione includono:

- Occhiali polarizzati: Questi occhiali riducono i riflessi perché permettono solo alla luce polarizzata lungo una certa direzione di attraversarli. - Brillio delle stelle: Le stelle appaiono scintillanti perché la luce che emettono subisce polarizzazione durante il suo percorso, mentre la luce riflessa dalla Luna e dai pianeti tende ad essere meno brillante.

La **polarizzazione dei fotoni** deriva direttamente dalla polarizzazione delle onde elettromagnetiche classiche. I fotoni ereditano la direzione di propagazione e la polarizzazione dell'onda da cui provengono. L'energia di un fotone è legata alla frequenza $\omega$ dell'onda attraverso la relazione:

$$E = h\omega,$$

dove $E$ è l'energia del fotone e $h$ è la costante di Planck.

### 3.4.6 I fotoni come qubits

Nei nostri esempi, i fotoni saranno le nostre *matite quantistiche*. Analogamente ai *bit-matita* utilizzati per rappresentare i bit classici, i fotoni ci serviranno per definire i **qubits**.

La polarizzazione di un fotone può essere utilizzata per rappresentare i due stati di un qubit. Possiamo definire due stati di polarizzazione ortogonali, ad esempio:

- $|0\rangle$: polarizzazione verticale - $|1\rangle$: polarizzazione orizzontale

Questa notazione ci permette di associare direttamente lo stato fisico del fotone al valore del qubit.

Strumenti come **QuantumSim** possono aiutare a visualizzare e comprendere il comportamento dei qubits basati su fotoni, senza necessità di proporne ancora l'uso diretto.

## 3.5 Fotoni e qubits

In questo paragrafo, approfondiremo il concetto di **qubit** e capiremo come realizzarlo utilizzando un sistema fisico quantistico, in particolare la polarizzazione di un fotone.

### 3.5.1 Confronto tra bits e qubits

Abbiamo visto che, in un sistema classico, i due possibili stati della matita (verticale e orizzontale) rappresentano un esempio di bit. Allo stesso modo, lo **spazio di tutti i possibili stati** in cui può trovarsi la polarizzazione di un fotone costituisce un esempio di qubit.

La relazione tra i due stati della matita e i valori 1 e 0 del bit è chiara. Misurando l'angolo che la matita forma rispetto a un certo piano, possiamo determinare il valore del bit associato. Una volta stabilita la convenzione che la matita verticale significa 1 e quella orizzontale significa 0, non è possibile alcun fraintendimento tra diversi osservatori, poiché la direzione verticale è definita utilizzando una direzione preferenziale: quella della forza di gravità.

Nel caso dei fotoni, invece, non abbiamo un fenomeno come la gravità per definire una direzione preferenziale, poiché la polarizzazione dei fotoni non ne è influenzata. Questo introduce delle differenze fondamentali tra bits e qubits.

### 3.5.2 La differenza nel concetto di stato

Tra qubits e bits c'è un'importante analogia: entrambi sono associati a sistemi fisici caratterizzati da due soli stati. Tuttavia, esistono differenze significative dovute al diverso significato attribuito alla parola *stato* nella fisica classica e in quella quantistica.

Nel contesto classico, lo stato di un sistema (come la posizione della matita) è ben definito e può essere misurato senza ambiguità. Nella meccanica quantistica, invece, il concetto di stato include la possibilità di sovrapposizione e indeterminazione, rendendo la descrizione più complessa.

### 3.5.3 Un esempio pratico: il bit-fotone

Per illustrare come utilizzare lo stato quantico della polarizzazione di un fotone per associare un qubit—che possiamo chiamare scherzosamente *bit-fotone*—presenteremo una breve storia nel prossimo paragrafo. Questa storia servirà a dare una visione d'insieme dei concetti presentati finora ed è necessaria per avvicinarci gradualmente allo studio della meccanica quantistica, che verrà approfondito nel capitolo 3.

Strumenti come **QuantumSim** possono aiutarci a visualizzare e comprendere le differenze tra gli stati classici e quantistici, facilitando l'apprendimento dei concetti chiave dell'informatica quantistica.

## 3.6 Il signor Rossi e il signor Ferrari: una classica storia di bits

Il signor Rossi è una persona a modo, abituata a non distinguersi troppo dagli altri. Ogni mattina, quando arriva al lavoro, dispone le sue matite per formare un codice binario che verrà letto pochi minuti dopo dal signor Ferrari, suo collega.

Al contrario di Rossi, il signor Ferrari è un tipo eccentrico e ama farsi notare. Invece di camminare come tutti gli altri, si sposta attaccato a due pertiche, mantenendo una posizione obliqua rispetto al suolo. Ama vedere il mondo da un altro punto di vista.

Quando, al mattino, si reca presso la scrivania del signor Rossi per leggere i bit rappresentati dalle matite, non si lascia confondere dal proprio punto di vista. Sa infatti che deve giudicare lo stato dei bit a partire dalla direzione in cui penzola la sua cravatta! Così, nonostante la sua posizione inclinata, riesce sempre a interpretare correttamente i bit di Rossi.

## 3.7 Il signor Magri e il signor Fabbri: una storia di qubits

Il signor Magri produce molti dati tutti i giorni, e non avendo modo di registrarli tutti in un archivio, deve inviarli a chi ne ha bisogno: il signor Fabbri, poi eliminarli. Il signor Magri abita lontano dal signor Fabbri. Ogni mattina si assicura di preparare un dispositivo polarizzatore per trasmettere dei fotoni polarizzati al signor Fabbri.
Il signor Magri è una persona scrupolosa e prepara i fotoni in modo che essi vengano prodotti con polarizzazione verticale $|1\rangle$ oppure orizzontale $|0\rangle$.
Con polarizzazione verticale ($|1\rangle$), egli intende un fotone che attraversi (senza essere fermato) un filtro *polaroid* con asse di polarizzazione disposto verticalmente, cioè nella direzione in cui agisce la gravità, mentre con polarizzazione orizzontale ($|0\rangle$) un fotone che attraversi un polaroid con asse orizzontale. Egli chiama l'insieme formato da $|1\rangle$ e $|0\rangle$ **base standard** per la polarizzazione dei fotoni.

### 3.7.1 Il primo incidente: dati recuperabili

Il signor Fabbri, ogni mattina, attende i fotoni del signor Magri, con i quali riceve informazioni importanti. Egli sa che i fotoni possono avere solo due possibili polarizzazioni, ma ha frainteso l'accordo con Magri e ha sistemato gli assi di polarizzazione dei suoi polaroid in modo errato: li ha invertiti tra di loro, scambiando così i fotoni con polarizzazione $|1\rangle$ con quelli a polarizzazione $|0\rangle$, e di conseguenza i qubit a 1 con i qubit a 0.

Un giorno, per puro caso, il signor Fabbri e il signor Magri si sentono al telefono e ne approfittano per discutere dei dati che Magri ha appena inviato e che Fabbri non ha ancora cancellato. I due si rendono conto che qualcosa non va e presto si accorgono che tutti i valori dei qubit sono invertiti.

Il signor Fabbri ha l'archivio con tutti i dati trasmessi dall'inizio e Magri lo tranquillizza dicendogli che sarà sufficiente trasformare tutti i qubit che sono a 0 al valore 1 e quelli che sono a 1 al valore 0. In questo modo, Fabbri può recuperare i dati corretti.

### 3.7.2 Il secondo incidente: dati non recuperabili

Dopo altre due settimane di trasmissione dei dati secondo la stessa procedura, il signor Fabbri si accorge che, inavvertitamente, ha ruotato di 45 gradi entrambi gli assi dei polarizzatori. Nonostante questo disallineamento, il suo sistema ha continuato a funzionare, registrando i due possibili valori 0 o 1.

Per sua fortuna, Fabbri riesce a risalire al momento dell'*incidente*, che capisce essere avvenuto durante le pulizie del suo laboratorio. Forte dell'esperienza avuta dal primo incidente, ripone fiducia nell'esistenza di una legge fisica o informatica che permetta di risalire ai valori corretti dei qubit partendo dai valori registrati con i polarizzatori ruotati di 45 gradi in senso orario.

Egli però non conosce questa legge e si rivolge al signor Magri, scoprendo suo malgrado che neanche lui la conosce. Fortunatamente, Magri ha tenuto copia dei dati trasmessi. Confrontando gli ultimi dati trasmessi con quelli ricevuti, si accorgono che non c'è correlazione: infatti, i fotoni trasmessi come $|1\rangle$ sono stati ricevuti a volte come $|1\rangle$ e altre come $|0\rangle$, in modo apparentemente casuale e con la stessa frequenza.

Il signor Magri e il signor Fabbri capiscono che non c'è modo di risalire ai dati corretti partendo dai dati registrati. Anche questa volta la situazione si risolve grazie alla scrupolosità di Magri, che aveva conservato per qualche giorno in più i dati. Tuttavia, i due si rendono conto che il loro sistema di trasmissione nasconde qualcosa di particolare.

### 3.7.3 Un'osservazione interessante

Il signor Fabbri è molto amareggiato e si autoaccusa di negligenza. Il signor Magri, invece, è rimasto colpito dall'accaduto e decide di sperimentare ulteriormente. Chiede allora a Fabbri di ruotare volontariamente i due polaroid di un angolo di soli 30 gradi. Così facendo, Magri trasmette una serie di fotoni di cui archivia il valore per confrontarli con quelli ricevuti da Fabbri.

Dal confronto emerge un risultato inaspettato: questa volta esiste una relazione tra i qubit ricevuti e quelli inviati, ma è solo una relazione statistica. I qubit ricevuti sono corretti nel 75% dei casi e errati nel restante 25%.

## 3.8 I signori Magri, Fabbri, Rossi e Ferrari si incontrano

I quattro signori—Magri, Fabbri, Rossi e Ferrari—si incontrano e si raccontano le rispettive esperienze. Comprendono che l'utilizzo della posizione delle matite per rappresentare i bit è assai diverso dall'utilizzo della polarizzazione dei fotoni.

Il signor Magri vuole capire cosa sta succedendo con i fotoni e prova a impostare una serie di ragionamenti partendo da quanto ha osservato finora:

- Un fotone verticale ($|1\rangle$) passa attraverso un polaroid polarizzato in modo verticale.
- Un fotone orizzontale ($|0\rangle$) passa attraverso un polaroid polarizzato in modo orizzontale.
- Un fotone verticale è fermato da un polaroid orizzontale.
- Un fotone orizzontale è fermato da un polaroid verticale.

Desidera esprimere il fatto che un fotone polarizzato verticalmente ha il 50% di probabilità di passare attraverso un polaroid inclinato di 45 gradi, come ha capito dal secondo incidente.

Sapendo che nell'universo non esiste una direzione preferenziale, pensa che l'effetto di un fotone a polarizzazione verticale su un polaroid obliquo (ruotato di 45 gradi in senso antiorario) debba essere lo stesso di un fotone a polarizzazione obliqua (ruotato di 45 gradi in senso orario) su un polaroid verticale. In pratica, decide di considerare la rotazione della polarizzazione del fotone rispetto all'asse del polaroid anziché fare il contrario. Questo perché la cosa importante è l'angolo tra i due e non l'angolo rispetto alla verticale, dato che la polarizzazione non è influenzata dalla gravità (come visto in precedenza).

### 3.8.1 La rappresentazione matematica degli stati

Per scrivere i due stati del fotone rispetto al polaroid ruotato, Magri immagina di ruotare nel piano cartesiano un vettore di lunghezza unitaria centrato nell'origine. Associa all'asse $x$ lo stato $|0\rangle$ e all'asse $y$ lo stato $|1\rangle$. La rotazione di 45 gradi in senso orario dello stato $|0\rangle$ di polarizzazione del fotone si scrive:

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

e quella della rotazione di 45 gradi in senso orario dello stato $|1\rangle$ del fotone come:

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

Il signor Magri postula che la probabilità $p$ che un fotone polarizzato a 45 gradi lungo la diagonale secondaria (come in figura) passi attraverso un polaroid con asse verticale sia data da:

$$p = \left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$$

Magri non sa spiegarsi il perché, ma capisce che c'è una relazione tra la radice di 2 trovata nella rotazione della polarizzazione del fotone e la probabilità del 50% ($\frac{1}{2}$) misurata sperimentalmente. Essendo un informatico, non è interessato alla polarizzazione dei fotoni in sé; ciò che cattura la sua attenzione è la relazione tra i qubit e lo stato di polarizzazione dei fotoni, ed è questo che vuole studiare.

Prima di continuare i suoi studi, capisce che le espressioni trovate sono importanti e decide di assegnare dei simboli agli stati individuati. Sceglie i due simboli $|0'\rangle$ e $|1'\rangle$ e li definisce come segue:

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

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

Si accorge quindi che può riscrivere gli stati $|0\rangle$ e $|1\rangle$ usando gli stati $|0'\rangle$ e $|1'\rangle$. Infatti:

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

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

Il signor Magri capisce che i qubit possono essere rappresentati anche attraverso questi due nuovi stati e che, quindi, $|0'\rangle$ e $|1'\rangle$ formano una **base alternativa** per rappresentare i qubit. Ha quindi intuito un concetto importante:

> *Lo stato di polarizzazione di un fotone è esprimibile rispetto a una base formata da due stati separati e indipendenti.*

Inoltre, lo stato del fotone può non coincidere con nessuno dei due stati di base, come è successo quando i fotoni con polarizzazione obliqua incidevano sui polaroid ad assi verticale e orizzontale.

Magri ha capito un'altra cosa molto importante:

> *Se la polarizzazione del fotone è parallela a uno degli assi dei polaroid, allora il risultato della misura è certo. Altrimenti, se la polarizzazione del fotone giace su una retta non parallela a nessuno dei due assi, il risultato è solo probabilistico.*

Questo concetto riflette una caratteristica fondamentale della meccanica quantistica: la probabilità intrinseca associata alle misure quando lo stato del sistema è una sovrapposizione degli stati di base.

Strumenti come **QuantumSim** possono aiutare a visualizzare questi fenomeni, permettendoci di esplorare come la rotazione degli stati e le diverse basi influenzino le probabilità di misura nei qubits, senza proporne ancora l'uso direttox.

## 3.9 Differenze tra bits e qubits

La storia del signor Magri e del signor Fabbri ci ha aperto gli occhi sulle differenze tra bits e qubits. Entrambi sfruttano lo stato di un sistema fisico per identificare una coppia di valori (0 e 1), e questa è la loro **analogia**. La **differenza** fondamentale è che, per i bits classici, c'è una corrispondenza *biunivoca* tra lo stato e il valore del bit, mentre per i qubits esiste una corrispondenza biunivoca tra la *misura* dello stato e il valore del qubit.

Nel caso classico, lo stato della matita identifica univocamente il valore del bit (verticale = 1, orizzontale = 0). Nel caso quantistico, invece, stati diversi possono portare probabilisticamente allo stesso valore del qubit. Non si può stabilire una corrispondenza biunivoca tra stato e valore tranne nei casi in cui lo stato di polarizzazione coincide con uno degli assi dei polaroid.

La misura dello stato di polarizzazione, tuttavia, porta sempre a uno dei due risultati possibili. Quindi, esiste una relazione biunivoca tra la misura dello stato e il valore del qubit.

Questa differenza è molto importante ed è ciò che caratterizza l'informatica quantistica rispetto a quella classica. È un concetto fondamentale che merita particolare attenzione e riflessione.

### 3.9.1 Perché questa differenza?

Nella fisica classica, non c'è differenza tra lo stato di un sistema e la sua misura. Ripensando all'esempio delle matite, si può immaginare di adottare la stessa convenzione a bordo di una **stazione spaziale** orbitante. In quel contesto, le matite possono assumere angoli arbitrari rispetto al riferimento della stazione. Sebbene una coppia di assi rimanga un valido sistema per definirne lo stato, si potrebbe obiettare che, analogamente a quanto accade nella meccanica quantistica, se una matita non è allineata lungo uno dei due assi, il suo stato è in una sovrapposizione di stati.

Questo non è falso, ma c'è una differenza importante e sostanziale con il concetto di stato quantistico. Dal punto di vista classico, anche se sulla stazione spaziale la matita può assumere una qualsiasi angolazione rispetto agli assi, rimane una relazione biunivoca tra stato e misura:

> *La misura dell'angolo che la matita forma con un dato asse può essere eseguita con precisione arbitraria senza modificare lo stato della matita.*

Anche su una stazione spaziale si potrebbero quindi usare i *bit-matita*, adottando la semplice convenzione che l'angolo da 0 a 45 gradi rappresenti il bit 0, mentre l'angolo da 45 a 90 gradi rappresenti il bit 1.

Per i sistemi quantistici, invece:

> **La relazione tra lo stato del sistema e il risultato della misura ha natura probabilistica.**

Come abbiamo visto, quando si misura lo stato di polarizzazione di un fotone, si possono ottenere solo due valori distinti. Prima della misura, il fotone può essere in una infinità di stati diversi tra loro. Il processo di misura *obbliga* il fotone ad assumere uno dei due stati che, per questo, sono detti *stati di base*.

Questo aspetto della meccanica quantistica è complesso da assimilare. Esistono anche interpretazioni diverse da questa, come l'*interpretazione di Copenaghen*, che offre una visione alternativa. Tuttavia, l'interpretazione di Copenaghen è la più diffusa ed è considerata l'interpretazione *ortodossa*.

## 3.10 Stato fisico del fotone e valore del qubit

Alcuni testi potrebbero tendere ad unificare il concetto di qubit con quello di stato fisico quantistico della polarizzazione del fotone. Indipendentemente dal modo in cui lo si vuole trattare formalmente, è sempre importante tenere a mente la distinzione esistente tra lo stato fisico prima e dopo l'esecuzione di una misura.

Ora che abbiamo introdotto il concetto di stato quantistico, dobbiamo chiarire bene qual è la relazione tra lo **stato fisico** e il **valore del qubit**.

Dato un fotone, si è visto che ad esso è associato un qubit legato al suo stato di polarizzazione, che possiamo indicare con la lettera $\psi$ o con il simbolo $|\psi\rangle$, come ha fatto il signor Magri.

Dal concetto di bit, ci aspettiamo che al qubit sia associato un valore binario, ovvero zero oppure uno.

Il valore binario associato al qubit non è il suo stato $|\psi\rangle$ in sé. Quindi, ci chiediamo in che modo sia legato ad esso.

Per determinare il valore di un qubit, è necessario sempre compiere una **misura**, nel caso in questione una misura della sua polarizzazione.

La misura *obbliga* lo stato (o funzione d'onda) $\psi$ a *collassare* in uno dei due stati che rappresentano la base scelta per eseguire la misura stessa.

Supponendo di aver eseguito la misura rispetto ai due assi dei polaroid, dopo la misura, lo stato $|\psi'\rangle$ del fotone potrà essere:

- $|\psi'\rangle = |0\rangle$ oppure
- $|\psi'\rangle = |1\rangle$

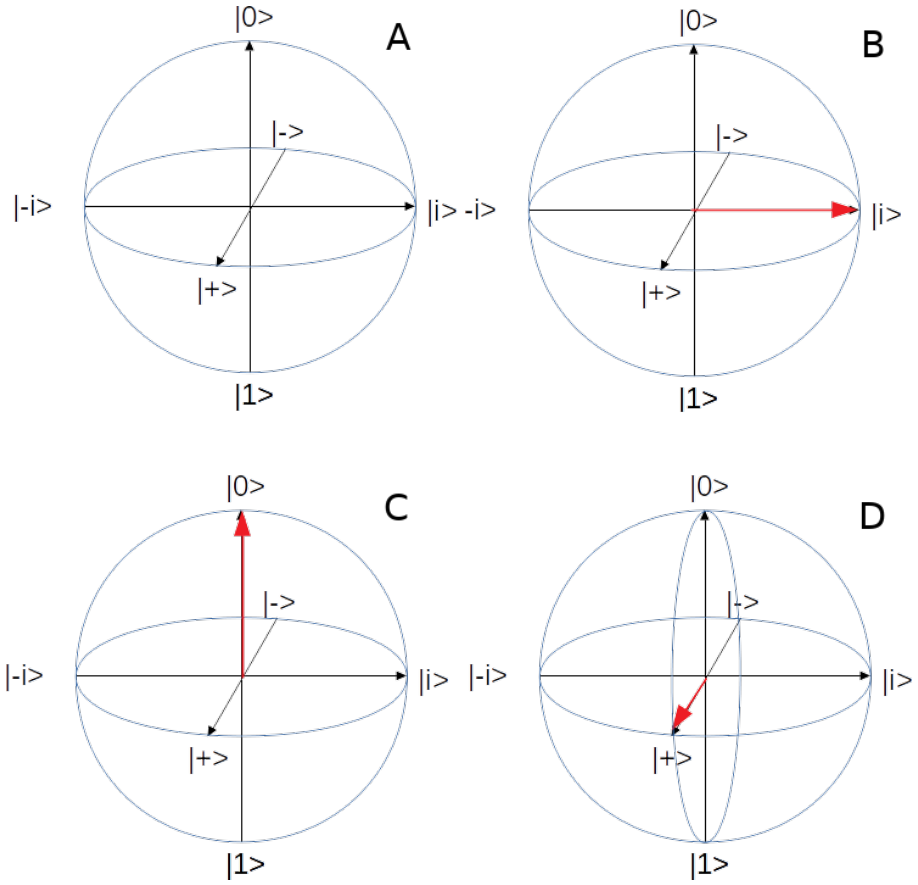



Figura 3.1: La figura mostra la sfera di Bloch, un sistema grafico per rappresentare lo stato di un qubit (**A**). Il vettore che giace lungo il semiasse positivo indica che il qubit è nello stato $|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ (**B**). Il qubit è nello stato $|0\rangle$ (**C**). Il qubit è nello stato $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ (**D**).

Questo **determina** il valore del qubit ad essere rispettivamente 0 oppure 1.

Lo stato di un qubit può essere rappresentato graficamente sulla *sfera di Bloch*, un sistema di mappatura per i sistemi quantistici a due livelli (vedi figura 3.1). Questo argomento avanzato è trattato in modo approfondito in *Qubits: principi fondamentali* degli stessi autori.

Strumenti come **QuantumSim** possono aiutare a visualizzare la sfera di Bloch e a comprendere meglio come lo stato del qubit cambi in funzione delle operazioni quantistiche, senza proporne ancora l'uso diretto.

## 3.11 Applicazioni con QuantumSim

In questa applicazione, utilizzeremo **QuantumSim** per visualizzare e comprendere la preparazione e la misurazione degli stati dei qubit utilizzando solo i gate Hadamard (H) e Pauli-X (X). Questo esempio semplice aiuterà a consolidare i concetti fondamentali della computazione quantistica presentati nelle sezioni precedenti.

### 3.11.1 Preparazione e Misurazione degli Stati con H e X

Per illustrare come i gate Hadamard e Pauli-X influenzano lo stato di un qubit, consideriamo due circuiti quantistici:

1. Circuito 1: Applica un gate Hadamard al qubit iniziale $|0\rangle$ e successivamente esegue una misurazione. 2. Circuito 2: Applica un gate Hadamard seguito da un gate Pauli-X al qubit iniziale $|0\rangle$ e successivamente esegue una misurazione.

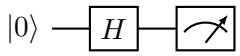



Figura 3.2: Circuito 1: Applicazione del gate Hadamard seguito da una misurazione

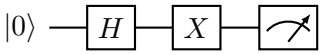



Figura 3.3: Circuito 2: Applicazione dei gate Hadamard e Pauli-X seguiti da una misurazione

Nel primo circuito (Figura 3.4), il gate Hadamard crea una sovrapposizione equa degli stati $|0\rangle$ e $|1\rangle$, preparando il qubit nello stato $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. La misurazione in questo stato dovrebbe restituire 0 o 1 con probabilità del 50% ciascuna.

Nel secondo circuito (Figura 3.5), dopo aver applicato il gate Hadamard, applichiamo un gate Pauli-X, che inverte lo stato del qubit: $|+\rangle$ diventa $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Anche in questo caso, la misurazione dovrebbe restituire 0 o 1 con probabilità del 50% ciascuna.

### 3.11.2 Codice QuantumSim per la Preparazione e Misurazione degli Stati

Di seguito è riportato un esempio di codice QuantumSim che implementa i circuiti descritti. Utilizzeremo le funzioni fornite da QuantumSim per applicare i gate Hadamard e Pauli-X, nonché per eseguire le misurazioni.

```
1
2 #include "quantum_sim.h"
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <math.h>
```

```c
#include <complex.h>
#include <time.h>

int circuit(void) {
    srand(time(NULL)); // Inizializza il generatore di numeri casuali

    // Inizializza lo stato quantistico con 1 qubit
    QubitState *state = initializeState(1);

    // Circuito 1: Applicazione del gate Hadamard seguito da una misurazione
    printf("Circuito 1: Applicazione del gate Hadamard seguito da una misurazione\n");
    initializeStateTo(state, 0); // Stato iniziale |0>
    applyHadamard(state, 0);      // Applica il gate Hadamard
    printf("Stato dopo H:\n");
    printState(state);
    MeasurementResult m1 = measure(state, 0); // Misura il qubit 0
    printf("Risultato della misurazione: %d\n\n", m1.result);

    // Circuito 2: Applicazione dei gate Hadamard e Pauli-X seguiti da una misurazione
    printf("Circuito 2: Applicazione dei gate Hadamard e Pauli-X seguiti da una misurazione\n");
    initializeStateTo(state, 0); // Reset allo stato |0>
    applyHadamard(state, 0);      // Applica il gate Hadamard
    applyX(state, 0);              // Applica il gate Pauli-X
    printf("Stato dopo H e X:\n");
    printState(state);
    MeasurementResult m2 = measure(state, 0); // Misura il qubit 0
    printf("Risultato della misurazione: %d\n\n", m2.result);

    // Libera la memoria allocata per lo stato quantistico
    freeState(state);

    return 0;
}
```

Listing 3.1: circuit3.c

### 3.11.3 Circuiti Quantistici Implementati

Di seguito sono riportati i circuiti quantistici corrispondenti alle operazioni eseguite nel codice QuantumSim:

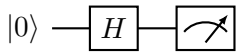



Figura 3.4: Circuito implementato per preparare e misurare lo stato $|+\rangle$

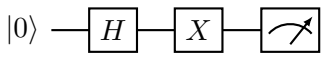



Figura 3.5: Circuito implementato per preparare e misurare lo stato $|-\rangle$

### 3.11.4 Analisi dei Risultati

Eseguendo il programma QuantumSim, otterrai risultati di misurazione che riflettono le probabilità teoriche associate ai circuiti implementati.

#### Circuito 1: Applicazione del Gate Hadamard

Il gate Hadamard applicato allo stato $|0\rangle$ prepara il qubit nello stato $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. La misurazione in questo stato dovrebbe restituire 0 o 1 con probabilità del 50% ciascuna.

```
Circuito 1: Applicazione del gate Hadamard seguito da una misurazione
Stato dopo H:
Stato 0: 0.707107 + 0.000000i | 0
Stato 1: 0.707107 + 0.000000i | 1
Risultato della misurazione: 0
```

#### Circuito 2: Applicazione dei Gate Hadamard e Pauli-X

Applicando un gate Pauli-X allo stato $|+\rangle$, otteniamo lo stato $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Anche in questo caso, la misurazione dovrebbe restituire 0 o 1 con probabilità del 50% ciascuna.

```
Circuito 2: Applicazione dei gate Hadamard e Pauli-X seguiti da una misurazione
Stato dopo H e X:
Stato 0: 0.707107 + 0.000000i | 0
Stato 1: -0.707107 + 0.000000i | 1
Risultato della misurazione: 1
```

### 3.11.5 Conclusioni

L'applicazione con QuantumSim ha dimostrato come i gate Hadamard e Pauli-X influenzano lo stato di un qubit durante la preparazione e la misurazione. La simulazione conferma che entrambi gli stati $|+\rangle$ e $|-\rangle$ presentano una probabilità del 50% di essere misurati come 0 o 1, evidenziando la natura probabilistica delle misure nei sistemi quantistici. Questo esempio semplice rafforza la comprensione delle differenze fondamentali tra bits classici e qubits quantistici, sottolineando l'importanza della sovrapposizione degli stati nella computazione quantistica.

### 3.11.6 Considerazioni sull'Uso di QuantumSim

QuantumSim si rivela uno strumento prezioso per visualizzare e sperimentare i concetti teorici della computazione quantistica. Attraverso la simulazione di circuiti quantistici semplici come quelli presentati, possiamo osservare direttamente le proprietà della sovrapposizione e della misura, facilitando l'apprendimento e la comprensione dei principi fondamentali che distinguono i qubits dai bits classici. Per approfondimenti futuri, è possibile esplorare circuiti più complessi che sfruttano ulteriori gate quantistici e fenomeni come l'entanglement, permettendo di ampliare le potenzialità della computazione quantistica.

## 3.12 Bibliografia ragionata

Il passaggio dai bits classici ai qubits quantistici rappresenta una rivoluzione nel modo in cui concepiamo l'informazione e la computazione. Per approfondire questi concetti, è fondamentale riferirsi a testi e articoli che hanno segnato tappe importanti nello sviluppo dell'informatica quantistica.

Un'opera fondamentale è il libro di Nielsen and Chuang [2010], *Quantum Computation and Quantum Information*, considerato una delle bibbie dell'informatica quantistica. Questo testo offre una trattazione completa dei principi teorici e delle applicazioni pratiche della computazione quantistica, inclusi i concetti di qubit, sovrapposizione degli stati, entanglement e algoritmi quantistici. È una risorsa indispensabile per chiunque voglia avere una comprensione approfondita del campo.

Per una comprensione dettagliata della meccanica quantistica, che è alla base dei qubits, il testo di Griffiths and Schroeter [2018], *Introduction to Quantum Mechanics*, fornisce un'esposizione chiara e accessibile dei principi fondamentali. Griffiths guida il lettore attraverso i concetti chiave, come la funzione d'onda, gli operatori, e il principio di indeterminazione di Heisenberg, che sono essenziali per comprendere come i qubits differiscano dai bits classici.

Un altro testo di riferimento è il lavoro di Merzbacher [1998], *Quantum Mechanics*, che offre una prospettiva storica e concettuale sulla meccanica quantistica. Merzbacher approfondisce i fondamenti teorici e le applicazioni, permettendo al lettore di comprendere le sottigliezze della teoria quantistica che sono cruciali per l'informatica quantistica.

Il concetto di sfera di Bloch, fondamentale per la rappresentazione grafica dello stato di un qubit, è trattato in modo dettagliato nel lavoro di Sakurai and Napolitano [2017], *Modern Quantum Mechanics*. Sakurai presenta una trattazione matematica rigorosa della meccanica quantistica, includendo discussioni sulla rappresentazione degli stati quantistici e sulle trasformazioni unitarie, che sono centrali nella manipolazione dei qubits.

Per quanto riguarda gli aspetti storici e concettuali dell'informatica quantistica, l'articolo di Deutsch [1985], *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer*, è fondamentale. Deutsch introduce l'idea che un calcolatore quanti-

stico possa eseguire compiti che un calcolatore classico non può, gettando le basi teoriche per la computazione quantistica.

Il lavoro di Feynman [1982], *Simulating Physics with Computers*, è un'altra pietra miliare nel campo. Feynman sostiene che i sistemi quantistici non possano essere simulati efficientemente dai computer classici, suggerendo che un computer quantistico sarebbe necessario per simulare la fisica quantistica. Questo articolo ha ispirato molti ricercatori a esplorare le possibilità offerte dalla computazione quantistica.

Per una prospettiva contemporanea sull'informazione quantistica, il libro di Preskill [2018], *Quantum Computing in the NISQ era and beyond*, offre un'analisi delle sfide attuali e future nel campo. Preskill introduce il concetto di dispositivi quantistici NISQ (Noisy Intermediate-Scale Quantum) e discute le prospettive dell'informatica quantistica nell'era attuale.

Infine, il lavoro di Shor [1997], *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, presenta l'algoritmo di Shor, che ha dimostrato il potenziale dei computer quantistici nel risolvere problemi complessi in modo esponenzialmente più veloce rispetto ai computer classici. Questo risultato ha avuto un impatto significativo sulla crittografia e ha stimolato ulteriori ricerche nel campo.

Queste fonti forniscono una base solida per comprendere le differenze tra bits classici e qubits quantistici, e per esplorare le implicazioni dell'informatica quantistica nella scienza e nella tecnologia.

## Transizione verso il Capitolo 4

Nel **Capitolo 4**, esploreremo i gate quantistici in dettaglio. Questi sono gli strumenti fondamentali per manipolare i qubits e implementare algoritmi quantistici. Analizzeremo la loro matematica, il loro impatto sulla Sfera di Bloch e il loro ruolo nel calcolo quantistico.

# Capitolo 4

# Principi della meccanica quantistica

## 4.1 Introduzione

In questo capitolo, gettiamo le fondamenta per una comprensione approfondita dei principi della meccanica quantistica, la teoria che descrive il comportamento della materia e dell'energia a livello microscopico. Dopo aver introdotto in modo informale i concetti chiave nel capitolo precedente, qui li riprenderemo in modo rigoroso, sviluppando il formalismo matematico necessario per affrontare le tematiche avanzate dell'informatica quantistica.

La meccanica quantistica rappresenta una rivoluzione nel modo in cui concepiamo il mondo naturale. Contrariamente alla fisica classica, che descrive il comportamento dei corpi macroscopici con leggi deterministiche, la meccanica quantistica introduce concetti come la sovrapposizione degli stati, l'indeterminazione e l'entanglement, che sfidano la nostra intuizione e richiedono un nuovo approccio concettuale.

In particolare, esploreremo:

- **Il contesto storico**: comprenderemo come le scoperte di inizio XX secolo abbiano portato alla necessità di una nuova teoria fisica.
- **Il formalismo matematico**: introdurremo gli strumenti matematici fondamentali, come gli spazi di Hilbert, gli operatori lineari e la notazione bra-ket di Dirac.
- **Il principio di sovrapposizione**: analizzeremo come gli stati quantistici possano combinarsi e quali sono le implicazioni di questo fenomeno.
- **Il ruolo della misura**: discuteremo come l'atto di misurare influenzi lo stato di un sistema quantistico, introducendo il concetto di collasso della funzione d'onda.
- **L'entanglement quantistico**: approfondiremo questo fenomeno fondamentale, che consente a particelle distanti di essere correlate in modi non spiegabili dalla fisica classica.

Questi concetti non solo sono essenziali per comprendere la meccanica quantistica in sé, ma costituiscono anche la base teorica su cui si costruisce l'informatica quantistica.

Ad esempio, l'entanglement è alla base di protocolli quantistici come la teleportazione quantistica e il quantum key distribution.

Il capitolo è strutturato in modo da guidare il lettore attraverso una progressione logica dei concetti, partendo dalle motivazioni storiche e arrivando al formalismo matematico necessario. Anche se alcune sezioni possono risultare impegnative dal punto di vista matematico, è importante non scoraggiarsi. L'obiettivo è fornire una comprensione profonda ma accessibile dei principi quantistici, che sarà preziosa nei capitoli successivi dedicati alle applicazioni nell'informatica.

Invitiamo il lettore a immergersi con curiosità e mente aperta in questo affascinante mondo, consapevoli che la meccanica quantistica non solo ha rivoluzionato la fisica, ma sta anche aprendo nuove frontiere nella tecnologia e nell'informazione.

## 4.2 L'importanza della visione di Dirac

Per comprendere e *accettare* la formulazione della meccanica quantistica, così come si presenta oggi, è importante partire dalla visione di chi l'ha concretamente formulata: P.A.M. Dirac (1957). A chi non è un fisico di professione, questo potrebbe sembrare strano: *"In fondo si tratta di formule che o sono giuste o sono sbagliate!"* Tuttavia, la fisica è la risposta alle domande che ci si pone, e la sua forma dipende dalle domande stesse e dal modo in cui vengono poste. Per questo motivo, studiare le risposte della fisica senza conoscere il pensiero di chi le ha formulate può essere molto complesso, specialmente se la formulazione non è intuitiva, come nel caso della meccanica quantistica.

Dirac scrive che l'obiettivo principale della fisica non è quello di fornire una raffigurazione della natura, ma di enunciare le leggi che governano i fenomeni conosciuti e che possono guidare alla scoperta di fenomeni nuovi. Si noti che Dirac afferma esattamente:

> ... the main object of physical science is not the provision of pictures, ...

evidenziando la **non necessità** di raffigurarsi il mondo quantistico. Dirac aveva un buon motivo per sostenere questo pensiero, poiché si era assunto la responsabilità di stravolgere la visione che la fisica aveva avuto fino ai primi anni '20.

## 4.3 Il contesto storico della meccanica quantistica

Il XX secolo era iniziato con diverse *novità* interessanti per la fisica. La formulazione delle leggi dell'elettromagnetismo aveva aperto le porte a nuove frontiere di studio e ricerca. All'inizio del secolo, **Ernest Rutherford** ipotizzò il modello planetario dell'atomo, **Louis de Broglie** propose la dualità onda-corpuscolo, **Max Planck** introdusse il quanto di azione che divenne la base per la meccanica quantistica, ed **Albert Einstein** spiegò l'effetto fotoelettrico basandosi sull'idea di fotone.

I fisici e le scoperte furono numerosi, ma ci concentriamo su questi per arrivare ad altri due scienziati fondamentali: **Erwin Schrödinger** e **Werner Heisenberg**. Furono tra

i primi a comprendere che le scoperte e le teorie dei primi del Novecento non potevano essere inserite nel *vecchio* paradigma classico newtoniano e che era necessario formulare un nuovo paradigma.

## 4.4 La meccanica ondulatoria di Schrödinger

Erwin Schrödinger sviluppò la **meccanica ondulatoria**, introducendo l'idea che le particelle subatomiche potessero essere descritte come onde. La sua famosa equazione d'onda permise di calcolare le probabilità di trovare una particella in una certa posizione nello spazio. Questo approccio era più intuitivo per i fisici dell'epoca, poiché si basava su concetti già noti legati alle onde.

## 4.5 La meccanica matriciale di Heisenberg

Parallelamente, Werner Heisenberg sviluppò la **meccanica matriciale**, un approccio più astratto che utilizzava matrici per descrivere le osservabili quantistiche. Sebbene inizialmente meno intuitiva, la meccanica matriciale si rivelò potente nel trattamento di sistemi quantistici complessi.

## 4.6 L'unificazione di Dirac

Paul Dirac riuscì a unificare questi due approcci, sviluppando una formulazione della meccanica quantistica che combinava gli aspetti migliori di entrambi. La sua teoria, elegante e coerente, divenne la base della meccanica quantistica moderna.

## 4.7 Una nuova visione della fisica

Questa breve introduzione ha lo scopo di preparare il lettore, magari poco familiare con la meccanica quantistica, a quanto seguirà. La meccanica quantistica può essere apprezzata più facilmente se si comprende che nasce dal pensiero di altri esseri umani. Non è semplicemente vera o falsa: è una teoria, un modo di pensare e di sperimentare, che si sta rivelando estremamente utile, tanto quanto lo è stata la meccanica newtoniana prima di essa.

## 4.8 Formulazione matematica del principio di sovrapposizione

Dobbiamo ora addentrarci nei dettagli della teoria quantistica. Nella trattazione che segue, si è cercato di estrapolare solo il necessario della teoria ai fini dell'introduzione all'informatica quantistica, attenendosi però rigorosamente al testo di Dirac, da cui nasce la

principale linea di interpretazione della meccanica quantistica, quella detta della *Scuola di Copenaghen*.

Il lettore sia rassicurato: non è necessario comprendere tutto perfettamente, ma è importante leggere questi capitoli per intero, in modo da non trovarsi impreparato nei capitoli dedicati all'informatica, dove i concetti e le idee qui esposte troveranno applicazione. Non si saltino poi le domande e ci si fermi a riflettere su di esse, consultando nel caso le risposte in fondo al testo.

## 4.9 Stato fisico

Consideriamo un sistema atomico composto da corpi puntiformi, ognuno con una specifica massa, carica elettrica, ecc. I corpi interagiscono tra loro attraendosi o respingendosi secondo certe leggi della fisica. Ci saranno vari possibili moti per ognuno di questi corpi che saranno in accordo con le leggi della fisica. Seguendo Dirac, chiameremo ognuno di questi possibili moti *stato* del sistema.

Nella meccanica classica, lo stato di un sistema può essere completamente definito misurando le tre coordinate $x$, $y$, $z$ e le tre componenti $v_x$, $v_y$, $v_z$ della velocità di ognuno dei corpi che costituisce il sistema. Questo risulta possibile perché la meccanica classica è stata storicamente applicata a sistemi di *grande massa* che coinvolgono sempre, almeno, decine di migliaia di atomi.

La meccanica quantistica si applica invece a sistemi composti anche da pochi o singoli atomi e alle particelle elementari. A tali dimensioni, si ha che:

> *I corpi non possiedono simultaneamente una posizione $x$ e una quantità di moto $p$ definita con precisione, ma esiste un'incertezza minima data da $\Delta x \, \Delta p_x \geq \frac{\hbar}{2}$.*

Quello appena enunciato è noto come il **principio di indeterminazione di Heisenberg**. A causa di questa incertezza, non è possibile definire il concetto di traiettoria per i corpi come atomi ed elettroni. Per questo motivo, il concetto di stato fisico **non può essere preso in prestito** dalla meccanica classica. Invece:

> *Lo stato di un sistema* piccolo *(quantistico) deve essere specificato usando meno variabili rispetto a uno classico o usando dati che sono più indefiniti.* (Dirac)

## 4.10 Probabilità e misura nella meccanica quantistica

Sia la meccanica ondulatoria che quella matriciale introducono un'ulteriore novità rispetto alla *visione* della natura che era stata indotta dalla meccanica classica.

In meccanica classica, conoscere lo stato di un sistema fisico significa poter prevedere il risultato di una misura con certezza. Si consideri, ad esempio, il sistema fisico costituito dalle sponde di un biliardo e dalle palle libere di rotolare sul tappeto. Se, a un certo istante, sono note le coordinate $x$, $y$ e le velocità $v_x$, $v_y$ di ogni pallina, **questo implica**

che il risultato di un esperimento che misurasse la posizione e la velocità delle palle darebbe esattamente le posizioni $x$, $y$ e le velocità $v_x$, $v_y$ note del sistema.

Nella meccanica classica questo è ovvio, in quanto, in linea di principio, non esiste differenza tra lo stato delle variabili di un sistema e le misure delle variabili del sistema, poiché le misure possono essere eseguite con precisione arbitraria.

Nel mondo atomico, le cose sono descritte diversamente. Quando si esegue una misura, il risultato dipende dallo stato, ma in modo **probabilistico**. Questo può sembrare inconsueto: ha lasciato perplessi molti fisici del passato e crea dubbi anche in molti del presente. Questo approccio alla fisica, comunque, porta a risultati coerenti con la sperimentazione e ha permesso di sviluppare molte nuove tecnologie, come appunto la computazione quantistica. Vale quindi la pena provare a capirlo.

## 4.11 Sovrapposizione di stati

**Nota**: In questo capitolo non verrà usata subito la notazione $|\psi\rangle$ per indicare uno stato quantistico già introdotta nel paragrafo precedente, perché l'argomento viene qui ripreso e spiegato dall'inizio in modo canonico.

### 4.11.1 Come si è arrivati alla sovrapposizione?

Come si è costruita una fisica in cui i risultati delle misure sono probabili anziché certi? L'idea sviluppata nella meccanica quantistica è semplice quanto non intuitiva. Si accetta che ogni stato fisico possa essere una sovrapposizione di altri stati fisici e che, quindi, un sistema possa trovarsi in una sovrapposizione di stati.

### 4.11.2 Idea classica

Per fare un esempio, pensiamo alle matite del signor Ferrari viste nel capitolo precedente. Dal punto di vista di Ferrari, esse sono oblique e quindi possono essere considerate come una sovrapposizione dello stato verticale e di quello orizzontale.

Rispetto al suolo, le matite possono essere rappresentate come:

$$\begin{aligned} bit\,0 &\to \hat{\mathbf{i}}\, l \\ bit\,1 &\to \hat{\mathbf{j}}\, l \end{aligned}$$

dove $l$ è la lunghezza della matita e $\hat{\mathbf{i}}$ e $\hat{\mathbf{j}}$ rappresentano i vettori direzionali dell'asse $x$ (orizzontale) e dell'asse $y$ (verticale) rispettivamente.

Dal punto di vista del signor Ferrari, che le guarda ruotato di $\frac{\pi}{4}$ (45 gradi in senso orario), la relazione tra il valore del bit e la loro direzione è la seguente:

$$\begin{aligned} bit\,0 &\to \frac{\hat{\mathbf{i}}'\, l}{\sqrt{2}} + \frac{\hat{\mathbf{j}}'\, l}{\sqrt{2}} \\ bit\,1 &\to \frac{\hat{\mathbf{j}}'\, l}{\sqrt{2}} - \frac{\hat{\mathbf{i}}'\, l}{\sqrt{2}} \end{aligned}$$

dove $\hat{\mathbf{i}}'$ e $\hat{\mathbf{j}}'$ rappresentano i vettori direzionali dell'asse $x'$ e dell'asse $y'$ solidali con il signor Ferrari.

Se consideriamo che la velocità della matita è nulla rispetto a Ferrari che la sta osservando, allora lo stato della matita è ben definito in senso classico. Lo stato della matita è lo stesso sia che lo si guardi dal sistema solidale con Ferrari, sia che lo si guardi dal sistema solidale con il terreno, cioè quello in cui l'asse $y$ è la verticale rispetto al suolo.

Quindi, lo stesso stato, per esempio il bit 0, può essere scritto sia come sovrapposizione di stati:

$$\left(\frac{\hat{\mathbf{i}}'\, l}{\sqrt{2}} + \frac{\hat{\mathbf{j}}'\, l}{\sqrt{2}}\right)$$

che come singolo stato:

$$(\hat{\mathbf{i}}\, l)$$

Quando il signor Ferrari esegue una misura dei *bit-matita* orizzontali, per esempio usando un goniometro, la misura darà sempre un risultato certo: 0 gradi nel sistema solidale con il terreno e 45 gradi nel sistema solidale con lui, indipendentemente dal sistema scelto per rappresentare lo stato.

In pratica, gli stati della meccanica classica possono essere espressi come combinazioni di altri stati, ma questo non influisce sulle misure delle variabili che caratterizzano lo stato.

### 4.11.3 Idea quantistica

L'idea quantistica di sovrapposizione di stati è simile in apparenza, ma con una differenza fondamentale. La introduciamo usando l'analogo quantistico dell'esempio delle matite: i fotoni del signor Fabbri.

Rappresentiamo il vettore di polarizzazione del fotone con il simbolo $\hat{\mathbf{e}}$. Ricordiamo che il signor Magri invia fotoni con polarizzazione $\hat{\mathbf{e}}$ verticale o orizzontale al signor Fabbri, il cui sistema di polaroid è ruotato di 45 gradi. Nel sistema del signor Magri, la polarizzazione orizzontale può essere espressa come:

$$\hat{\mathbf{e}} = \hat{\mathbf{i}}$$

mentre quella verticale come:

$$\hat{\mathbf{e}} = \hat{\mathbf{j}}$$

La polarizzazione del fotone $\hat{\mathbf{e}} = \hat{\mathbf{i}}$ inviato dal signor Magri, nel sistema del signor Fabbri è invece vista come:

$$\hat{\mathbf{e}} = \frac{1}{\sqrt{2}}\left(\hat{\mathbf{i}}' + \hat{\mathbf{j}}'\right)$$

dove $\hat{\mathbf{i}}'$ e $\hat{\mathbf{j}}'$ sono le direzioni degli assi ruotati dei suoi polaroid.

Qui, l'analogia tra meccanica quantistica e meccanica classica finisce.

### 4.11.4 Il collasso della funzione d'onda

L'aspetto imprevisto e **caratterizzante** della fisica quantistica è che la conoscenza dello stato del sistema fisico non implica la capacità di predire il risultato di una misura con certezza.

Supponiamo che il signor Magri prepari un fotone nello stato:

$$\hat{\mathbf{e}} = \hat{\mathbf{i}}$$

e che, prima di inviarlo, avverta il signor Fabbri dello stato di polarizzazione in cui giungerà il fotone.

Il signor Fabbri attende il fotone ed è pronto a misurarne lo stato nel suo sistema inclinato rispetto a quello di Magri. Tuttavia, il risultato della misura sarà *casualmente* o:

$$\hat{\mathbf{e}} = \hat{\mathbf{i}}'$$

oppure:

$$\hat{\mathbf{e}} = \hat{\mathbf{j}}'$$

perdendo traccia dell'una o dell'altra componente. Una volta eseguita la misura, lo stato *collassa* in una delle due direzioni $\hat{\mathbf{e}} = \hat{\mathbf{i}}'$ o $\hat{\mathbf{e}} = \hat{\mathbf{j}}'$, perdendo traccia dello stato di *sovrapposizione*:

$$\frac{1}{\sqrt{2}}\left(\hat{\mathbf{i}}' + \hat{\mathbf{j}}'\right)$$

Questo fenomeno, noto come **collasso della funzione d'onda**, è una caratteristica peculiare della meccanica quantistica.

Strumenti come **QuantumSim** possono aiutare a visualizzare questo processo di sovrapposizione e collasso, permettendo di comprendere meglio come le misure influenzino lo stato quantistico di un sistema, senza proporne ancora l'uso diretto.

1. $\hat{\mathbf{e}} = \dfrac{1}{\sqrt{2}}\left(\hat{\mathbf{i}} + \hat{\mathbf{j}}\right)$
2. $\hat{\mathbf{e}} = \hat{\mathbf{i}}$
3. $\hat{\mathbf{e}} = \hat{\mathbf{i}}$ *oppure* $\hat{\mathbf{e}} = \hat{\mathbf{j}}$ con uguale probabilità

## 4.12 Stato e probabilità

Dal paragrafo precedente, possiamo trarre alcune considerazioni riguardo alla misura dello stato di polarizzazione dei fotoni, che hanno validità generale nella teoria quantistica.

### 4.12.1 Espressione dello stato quantistico e probabilità

Rispetto a un certo strumento di misura, lo stato di polarizzazione di un fotone può essere espresso come combinazione lineare delle due direzioni $\hat{\mathbf{i}}$ e $\hat{\mathbf{j}}$. Con combinazione lineare si intende un'espressione del tipo:

$$\hat{\mathbf{e}} = \alpha\,\hat{\mathbf{i}} + \beta\,\hat{\mathbf{j}}$$

dove $\alpha$ e $\beta$ sono numeri complessi, identificati da un modulo (la loro lunghezza) e una fase (un angolo). Se lo stato di polarizzazione $\hat{\mathbf{e}}$ coincide con una delle due direzioni, cioè:

$$\hat{\mathbf{e}} = \hat{\mathbf{i}}$$

oppure

$$\hat{\mathbf{e}} = \hat{\mathbf{j}}$$

la misura risulterà rispettivamente e deterministicamente $\hat{\mathbf{i}}$ oppure $\hat{\mathbf{j}}$.

Lo stato di un sistema non cambia se viene moltiplicato per un numero complesso $\alpha$ di modulo 1 (cioè lunghezza pari a 1 e angolo qualsiasi):

$$\hat{\mathbf{e}} = \hat{\mathbf{i}} = \alpha\,\hat{\mathbf{i}}$$

oppure

$$\hat{\mathbf{e}} = \hat{\mathbf{j}} = \alpha\,\hat{\mathbf{j}}$$

Si dice quindi che lo stato $\hat{\mathbf{e}}$ è definito a meno di una costante moltiplicativa di modulo 1. Questo significa che **non c'è differenza fisica**[1] tra lo stato $\hat{\mathbf{i}}$ e lo stato $\alpha\,\hat{\mathbf{i}}$ se il numero complesso $\alpha$ ha modulo 1.

Se lo stato del fotone è una combinazione lineare degli stati base $\hat{\mathbf{i}}$ e $\hat{\mathbf{j}}$, cioè:

$$\hat{\mathbf{e}} = \alpha\,\hat{\mathbf{i}} + \beta\,\hat{\mathbf{j}}$$

il risultato della misura potrà essere *indeterminatamente* sia $\hat{\mathbf{i}}$ che $\hat{\mathbf{j}}$. La probabilità di ottenere una delle due misure non è necessariamente uguale alla probabilità di ottenere l'altra: la probabilità di ottenere $\hat{\mathbf{i}}$ è pari a $|\alpha|^2$, mentre quella di ottenere $\hat{\mathbf{j}}$ è pari a $|\beta|^2$.

Si noti che è il rapporto tra $\alpha$ e $\beta$ che **ha significato fisico**, in quanto determina la probabilità di ottenere lo stato $\hat{\mathbf{i}}$ o lo stato $\hat{\mathbf{j}}$ come risultato di una misura.

### 4.12.2 La base di uno stato quantistico

Vediamo ora una considerazione di cui è difficile sopravvalutare l'importanza. Se la somma $|\alpha|^2 + |\beta|^2$ è uguale a 1, allora significa che, indipendentemente da ogni altra variabile che

[1] Vedi *Qubits: principi fondamentali* degli stessi autori.

può caratterizzare il sistema, la polarizzazione dovrà sempre essere o lungo $\hat{\mathbf{i}}$, o lungo $\hat{\mathbf{j}}$. In questo senso, diremo che $\{\hat{\mathbf{i}}, \hat{\mathbf{j}}\}$ costituiscono una **base** per lo stato di un fotone.

La teoria impostata fino a qui fornisce gli strumenti di calcolo per determinare la probabilità di ottenere un certo risultato misurando **un singolo** fotone che si trovi nel generico stato $\hat{\mathbf{e}}$.

Ma come si generalizza questo risultato quando lo stato fisico è relativo non a uno, ma a due o più fotoni?

Per estendere questi concetti a più fotoni (e in generale a più particelle), è necessario introdurre il calcolo tensoriale.

## 4.13 Spazi vettoriali e prodotti tensoriali

Questo paragrafo ha un taglio più matematico, ma per comprendere completamente i capitoli centrali dedicati all'informatica quantistica, è utile afferrare a fondo sia i concetti sia il formalismo qui introdotti. Comunque, se questo risultasse troppo complesso, non scoraggiarti: il resto può essere compreso **a livello intuitivo**, anche senza un rigoroso formalismo matematico.

### 4.13.1 Base di uno spazio vettoriale

I tre versori $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ e $\hat{\mathbf{u}}$ sono una base dello spazio vettoriale $\mathbb{R}^3$. Ciò significa che qualunque vettore $\mathbf{v}$ dello spazio può essere scritto come:

$$\mathbf{v} = \alpha\,\hat{\mathbf{i}} + \beta\,\hat{\mathbf{j}} + \gamma\,\hat{\mathbf{u}}$$

dove $\alpha$, $\beta$ e $\gamma$ sono numeri reali.

**Nota:** normalmente, i versori si indicano con le lettere $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ e $\hat{\mathbf{k}}$, ma in questo testo viene usata la lettera $\hat{\mathbf{u}}$ al posto della $\hat{\mathbf{k}}$ per motivi tipografici. Una forma alternativa per indicare la base di $\mathbb{R}^3$ è utilizzare gli elementi $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ e $\boldsymbol{e}_3$. Questa notazione ha il vantaggio di poter essere usata anche per spazi vettoriali di dimensione superiore ($\mathbb{R}^4$, $\mathbb{R}^5$, ...), poiché gli assi dello spazio sono indicati con un indice numerico che può essere esteso a piacere.

Gli elementi $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_n$ costituiscono una base per un generico spazio vettoriale $V$ di dimensione $n$.

### 4.13.2 Spazio duale e base duale

Introduciamo ora il concetto di elemento duale e di spazio vettoriale duale. Questo è un concetto nuovo per chi non ha studi specifici di algebra lineare avanzata.

Accanto agli elementi $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_n$, introduciamo gli elementi $\boldsymbol{e}^1, \boldsymbol{e}^2, \boldsymbol{e}^3, \ldots, \boldsymbol{e}^n$, detti elementi duali dei primi.

Tra questi due insiemi di elementi esiste una relazione espressa come segue:

$$\boldsymbol{e}^i\,(\boldsymbol{e}_j) = \delta^i_j$$

dove $\delta^i_j$ è il delta di Kronecker, che vale 1 se $i = j$ e 0 altrimenti.

Gli elementi $\boldsymbol{e}^1, \boldsymbol{e}^2, \ldots, \boldsymbol{e}^n$ formano la base di uno spazio vettoriale che chiamiamo **spazio duale** di $V$.

### 4.13.3 Il prodotto tensoriale

Introduciamo ora una nuova operazione, indicata con il simbolo $\otimes$, chiamata **prodotto tensoriale**.

Consideriamo due spazi vettoriali $V$ e $W$, con basi rispettivamente $\boldsymbol{e}_i$ e $\boldsymbol{f}_j$. Il prodotto tensoriale tra due elementi, uno di $V$ e uno di $W$, è definito come:

$$\boldsymbol{e}_i \otimes \boldsymbol{f}_j$$

Quando non crea ambiguità, possiamo scrivere:

$$\boldsymbol{e}_i \otimes \boldsymbol{f}_j = \boldsymbol{e}_i \boldsymbol{f}_j$$

Il prodotto tensoriale degli spazi $V$ e $W$ è dato da:

$$V \otimes W = \text{span}\,\{\boldsymbol{e}_i \boldsymbol{f}_j \,|\, i = 1, \ldots, n;\, j = 1, \ldots, m\}$$

Familiarizzare con il prodotto tensoriale è fondamentale perché permette di comprendere a fondo il concetto di **entanglement**, uno dei fondamenti dell'informatica quantistica.

## 4.14 Prodotto tensoriale tra due vettori

Se $\mathbf{v}$ e $\mathbf{w}$ sono due vettori, il primo appartenente allo spazio $V$ e il secondo allo spazio $W$ (che per semplicità possiamo assumere essere lo stesso spazio, cioè $V = W$), il loro prodotto tensoriale è dato da:

$$\mathbf{v} \otimes \mathbf{w} = \left( \sum_{i=1}^{n} v_i \boldsymbol{e}_i \right) \otimes \left( \sum_{j=1}^{n} w_j \boldsymbol{f}_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} v_i w_j \, \boldsymbol{e}_i \boldsymbol{f}_j$$

Dove:

$$\mathbf{v} = v_1 \boldsymbol{e}_1 + v_2 \boldsymbol{e}_2 + \cdots + v_n \boldsymbol{e}_n$$

$$\mathbf{w} = w_1 \boldsymbol{f}_1 + w_2 \boldsymbol{f}_2 + \cdots + w_n \boldsymbol{f}_n$$

Vediamo un esempio del prodotto tensoriale tra due vettori di $\mathbb{R}^2$.

Siano $\mathbf{v} = 3\boldsymbol{e}_1 + 5\boldsymbol{e}_2$ e $\mathbf{w} = 2\boldsymbol{f}_1 + 4\boldsymbol{f}_2$ due vettori di $\mathbb{R}^2$. Il loro prodotto tensoriale è:

$$\mathbf{v} \otimes \mathbf{w} = (3\boldsymbol{e}_1 + 5\boldsymbol{e}_2) \otimes (2\boldsymbol{f}_1 + 4\boldsymbol{f}_2) = 3 \times 2\, \boldsymbol{e}_1 \boldsymbol{f}_1 + 3 \times 4\, \boldsymbol{e}_1 \boldsymbol{f}_2 + 5 \times 2\, \boldsymbol{e}_2 \boldsymbol{f}_1 + 5 \times 4\, \boldsymbol{e}_2 \boldsymbol{f}_2$$

$$= 6\,\boldsymbol{e}_1\boldsymbol{f}_1 + 12\,\boldsymbol{e}_1\boldsymbol{f}_2 + 10\,\boldsymbol{e}_2\boldsymbol{f}_1 + 20\,\boldsymbol{e}_2\boldsymbol{f}_2$$

Da questa espressione si vede che il prodotto tensoriale dei due vettori genera un nuovo vettore, espresso rispetto agli elementi:

$$\boldsymbol{e}_1\boldsymbol{f}_1, \quad \boldsymbol{e}_1\boldsymbol{f}_2, \quad \boldsymbol{e}_2\boldsymbol{f}_1, \quad \boldsymbol{e}_2\boldsymbol{f}_2$$

Questi elementi costituiscono la base dello spazio $\mathbb{R}^2 \otimes \mathbb{R}^2$, che ha dimensione 4.

Abbiamo visto che un qubit è associato a uno spazio di dimensione 2, dove ciascuno dei due assi rappresenta una direzione di polarizzazione. Per rappresentare due qubit, è necessario considerare il prodotto tensoriale tra due spazi di dimensione 2, analogamente a quanto fatto qui.

Strumenti come **QuantumSim** possono aiutare a visualizzare e comprendere meglio questi concetti complessi, offrendo simulazioni interattive delle operazioni sui qubit, senza proporne ancora l'uso diretto.

### 4.14.1 Stato di 2 qubit

Come lo stato di un qubit è rappresentabile per mezzo di un vettore nello spazio $V$, lo stato di 2 qubit è rappresentabile con un vettore nello spazio tensoriale ottenuto dal prodotto $V \otimes V$, che ha dimensione 4, cioè $2 \times 2$.

Questo aspetto della teoria quantistica può sorprendere perché, nella teoria classica, lo spazio per descrivere due punti materiali non è il prodotto tensoriale $\mathbb{R}^3 \otimes \mathbb{R}^3$, ma il prodotto cartesiano $\mathbb{R}^3 \times \mathbb{R}^3$. In un sistema classico, infatti, la posizione di un singolo punto materiale è descritta da un vettore di $\mathbb{R}^3$, e la posizione di due punti materiali è descritta da un vettore di $\mathbb{R}^6$, che ha dimensione $3 + 3$ e non $3 \times 3$.

La differenza rispetto al caso classico non risulta eclatante nel caso di due soli qubit, perché $2 \times 2$ è uguale a $2 + 2$, ma diventa evidente se si considera un sistema formato da un numero elevato $n$ di qubit, che viene descritto in uno spazio di dimensione $2 \times 2 \times \cdots \times 2$, per $n$ volte, quindi $2^n$.

Riassumendo: nella visione classica, $n$ corpi sono descritti da un vettore dello spazio $\mathbb{R}^3 \times \mathbb{R}^3 \times \cdots \times \mathbb{R}^3$, per $n$ volte, quindi uno spazio con $n \times 3$ dimensioni (o assi). Nella teoria quantistica, lo stato di un sistema di $n$ qubit è un vettore nello spazio $V \otimes V \otimes \cdots \otimes V$, moltiplicato per $n$ volte, che ha $2^n$ dimensioni.

La differenza nel calcolo delle dimensioni dello spazio usato per descrivere un sistema quantistico rispetto a uno classico, quando si passa da una a $n$ particelle, si spiega con il ruolo diverso che gioca lo stato fisico nei due diversi *paradigmi teorici*. Infatti, come abbiamo visto, allo stato quantistico di un qubit è associato il calcolo della probabilità di ottenere un dato risultato in conseguenza di una certa misura, e il prodotto tensoriale è l'operazione che permette di scalare da 1 a $n$ qubit coerentemente con la statistica.

Per questo motivo, lo stato di $n$ qubit è descritto dal loro prodotto tensoriale.

Nel seguito, tratteremo sempre esempi con al massimo 3 qubit.

## 4.15 Stati entangled

Come è stato scritto nell'introduzione, non è necessario conoscere la fisica quantistica per apprezzare e sviluppare l'informatica quantistica, ma per apprezzarne appieno il *dense coding*, la *quantum teleportation* e il *no cloning principle*, è necessario sapere cosa sia l'*entanglement*.

Una descrizione puramente *pittorica* sarebbe stata più semplice, ma poco utile ai fini informatici.

Gli sforzi matematici per arrivare fin qui saranno ora premiati: infatti, arrivati a questo punto, si avrà piena consapevolezza del perché nella meccanica quantistica è stato introdotto l'*entanglement* e cosa esso rappresenti veramente.

Abbiamo appena visto come si presenta il prodotto tensoriale tra due vettori.

Ogni vettore ottenuto dal prodotto di due vettori appartenenti allo spazio $V$ dà luogo a un altro vettore nello spazio vettoriale $V \otimes V$, ma bisogna fare attenzione, perché *non è detto* il contrario. Esistono infatti elementi dello spazio $V \otimes V$ che non possono essere scritti come il prodotto tra due vettori $\mathbf{v}$ e $\mathbf{w}$ dello spazio $V$: in questo caso, vedremo che gli stati associati a tali elementi si dicono **entangled**.

Per vedere un esempio, consideriamo la seguente espressione:

$$\psi = a\,\boldsymbol{e}_1 \otimes \boldsymbol{f}_1 + b\,\boldsymbol{e}_2 \otimes \boldsymbol{f}_2$$

e ci chiediamo se sia possibile trovare due vettori $\mathbf{v}$ e $\mathbf{w}$ tali che:

$$\mathbf{v} \otimes \mathbf{w} = \psi = a\,\boldsymbol{e}_1 \otimes \boldsymbol{f}_1 + b\,\boldsymbol{e}_2 \otimes \boldsymbol{f}_2$$

Per rispondere, sviluppiamo il prodotto al primo membro ($\mathbf{v} \otimes \mathbf{w}$), ottenendo:

$$\begin{aligned}\mathbf{v} \otimes \mathbf{w} &= (v_1\boldsymbol{e}_1 + v_2\boldsymbol{e}_2) \otimes (w_1\boldsymbol{f}_1 + w_2\boldsymbol{f}_2) \\ &= v_1w_1\,\boldsymbol{e}_1 \otimes \boldsymbol{f}_1 + v_1w_2\,\boldsymbol{e}_1 \otimes \boldsymbol{f}_2 + v_2w_1\,\boldsymbol{e}_2 \otimes \boldsymbol{f}_1 + v_2w_2\,\boldsymbol{e}_2 \otimes \boldsymbol{f}_2\end{aligned}$$

Affinché questa espressione sia uguale a $\psi$, i coefficienti dei termini $\boldsymbol{e}_1 \otimes \boldsymbol{f}_2$ e $\boldsymbol{e}_2 \otimes \boldsymbol{f}_1$ devono essere nulli, cioè:

$$v_1w_2 = 0 \quad \text{e} \quad v_2w_1 = 0$$

Questo implica che o $v_1 = 0$ o $w_2 = 0$, e che o $v_2 = 0$ o $w_1 = 0$. Ma se, ad esempio, $v_1 = 0$ e $v_2 = 0$, allora $\mathbf{v}$ sarebbe il vettore nullo, e il prodotto tensoriale sarebbe nullo. Se invece $w_1 = 0$ e $w_2 = 0$, allora $\mathbf{w}$ sarebbe il vettore nullo.

Quindi, per soddisfare entrambe le condizioni senza annullare completamente $\mathbf{v}$ o $\mathbf{w}$, dovremmo avere $v_1 = 0$ e $w_1 = 0$, o $v_2 = 0$ e $w_2 = 0$. In entrambi i casi, almeno uno dei

prodotti $v_1 w_1$ o $v_2 w_2$ sarebbe nullo, rendendo impossibile ottenere i termini $a\,\boldsymbol{e}_1 \otimes \boldsymbol{f}_1$ e $b\,\boldsymbol{e}_2 \otimes \boldsymbol{f}_2$ simultaneamente (a meno che uno dei coefficienti $a$ o $b$ sia zero).

Abbiamo quindi dimostrato che esistono stati di due qubit *entangled*, cioè che non possono essere scritti come il prodotto di due qubit considerati singolarmente.

Questo risultato è assolutamente diverso da quanto accade per i bit classici, dove lo stato di un sistema di bit è sempre dato dal prodotto cartesiano degli stati dei singoli bit.

Lo stato di due qubit che non possa essere scritto come il prodotto dei singoli qubit è detto essere **entangled**.

Questo è un risultato che si applica a tutta la meccanica quantistica (elettroni, protoni, ecc.), non solo ai qubit, ma qui focalizziamo l'attenzione solo sugli aspetti legati all'*entanglement* tra qubit.

## 4.16 Prodotto tensoriale tra due duali

Quanto visto per i vettori si applica anche ai vettori **duali**.

Per $\mathbf{v}^*$ e $\mathbf{w}^*$ appartenenti agli spazi duali di $V$ e $W$, scriveremo:

$$\mathbf{v}^* \otimes \mathbf{w}^* = v_1^* w_1^*\, \boldsymbol{e}_1^* \otimes \boldsymbol{f}_1^* + v_1^* w_2^*\, \boldsymbol{e}_1^* \otimes \boldsymbol{f}_2^* + v_2^* w_1^*\, \boldsymbol{e}_2^* \otimes \boldsymbol{f}_1^* + v_2^* w_2^*\, \boldsymbol{e}_2^* \otimes \boldsymbol{f}_2^* + \dots$$

## 4.17 Prodotto tensoriale tra vettori e duali

È importante anche definire il prodotto tra vettori e **duali**.

Per $\mathbf{v}$ appartenente allo spazio $V$ e $\mathbf{w}^*$ appartenente al duale di $W$, scriveremo:

$$\mathbf{v} \otimes \mathbf{w}^* = v_1 w_1^*\, \boldsymbol{e}_1 \otimes \boldsymbol{f}_1^* + v_1 w_2^*\, \boldsymbol{e}_1 \otimes \boldsymbol{f}_2^* + v_2 w_1^*\, \boldsymbol{e}_2 \otimes \boldsymbol{f}_1^* + v_2 w_2^*\, \boldsymbol{e}_2 \otimes \boldsymbol{f}_2^* + \dots$$

## 4.18 Matrici e tensori

**Nota bene**: La trattazione completa ed esaustiva di questo argomento deve essere condotta su un testo dedicato come il Landau o il Dirac. Qui si cercherà di delineare il minimo formalismo necessario per comprendere gli aspetti dell'informatica quantistica, facendo comunque attenzione a non semplificare eccessivamente in modo da non introdurre concetti falsi o ambigui.

Nella meccanica quantistica, gli stati fisici sono rappresentati dai vettori. Le trasformazioni fisiche che avvengono sugli stati, per esempio l'azione di un campo elettrico su un elettrone, sono rappresentate come **operatori** che agiscono sugli stati fisici trasformandoli. Dal punto di vista matematico, gli operatori della fisica sono visti come dei tensori.

Consideriamo ancora un fotone descritto dal vettore di polarizzazione $\hat{\mathbf{e}} = \alpha\,\boldsymbol{e}_1 + \beta\,\boldsymbol{e}_2$. Il vettore può essere ruotato in senso orario di un angolo $\theta$, per esempio $\theta = \frac{\pi}{4}$, mediante l'operatore $\boldsymbol{R}$:

$$\hat{\mathbf{e}}' = \boldsymbol{R}\,\hat{\mathbf{e}}$$

per ottenere il nuovo vettore $\hat{\mathbf{e}}'$.

L'espressione usata per definire la rotazione è puramente formale e non dice nulla sulla relazione esistente tra $\hat{\mathbf{e}}$ e $\hat{\mathbf{e}}'$. Per concretizzare la rotazione, dobbiamo dare una forma esplicita a $\boldsymbol{R}$. Ad esempio, consideriamo la seguente espressione per l'operatore $\boldsymbol{R}$:

$$\boldsymbol{R} = \cos\theta\,\boldsymbol{e}_1 \otimes \boldsymbol{e}_1^* + \sin\theta\,\boldsymbol{e}_1 \otimes \boldsymbol{e}_2^* - \sin\theta\,\boldsymbol{e}_2 \otimes \boldsymbol{e}_1^* + \cos\theta\,\boldsymbol{e}_2 \otimes \boldsymbol{e}_2^*$$

Ricordando che gli elementi duali agiscono sui vettori come funzionali lineari (cioè $\boldsymbol{e}_i^*(\boldsymbol{e}_j) = \delta_{ij}$), vediamo che applicando $\boldsymbol{R}$ a $\hat{\mathbf{e}}$ si ottiene l'espressione per la rotazione del vettore $\hat{\mathbf{e}}$ di un angolo $\theta$:

$$\begin{aligned}
\boldsymbol{R}\,\hat{\mathbf{e}} &= \left(\cos\theta\,\boldsymbol{e}_1 \otimes \boldsymbol{e}_1^* + \sin\theta\,\boldsymbol{e}_1 \otimes \boldsymbol{e}_2^* - \sin\theta\,\boldsymbol{e}_2 \otimes \boldsymbol{e}_1^* + \cos\theta\,\boldsymbol{e}_2 \otimes \boldsymbol{e}_2^*\right)(\alpha\,\boldsymbol{e}_1 + \beta\,\boldsymbol{e}_2) \\
&= \alpha\left(\cos\theta\,\boldsymbol{e}_1 \otimes \boldsymbol{e}_1^*(\boldsymbol{e}_1) + \sin\theta\,\boldsymbol{e}_1 \otimes \boldsymbol{e}_2^*(\boldsymbol{e}_1) - \sin\theta\,\boldsymbol{e}_2 \otimes \boldsymbol{e}_1^*(\boldsymbol{e}_1) + \cos\theta\,\boldsymbol{e}_2 \otimes \boldsymbol{e}_2^*(\boldsymbol{e}_1)\right) \\
&\quad + \beta\left(\cos\theta\,\boldsymbol{e}_1 \otimes \boldsymbol{e}_1^*(\boldsymbol{e}_2) + \sin\theta\,\boldsymbol{e}_1 \otimes \boldsymbol{e}_2^*(\boldsymbol{e}_2) - \sin\theta\,\boldsymbol{e}_2 \otimes \boldsymbol{e}_1^*(\boldsymbol{e}_2) + \cos\theta\,\boldsymbol{e}_2 \otimes \boldsymbol{e}_2^*(\boldsymbol{e}_2)\right) \\
&= \alpha\left(\cos\theta\,\boldsymbol{e}_1 - \sin\theta\,\boldsymbol{e}_2\right) + \beta\left(\sin\theta\,\boldsymbol{e}_1 + \cos\theta\,\boldsymbol{e}_2\right) \\
&= \left(\alpha\cos\theta + \beta\sin\theta\right)\boldsymbol{e}_1 + \left(-\alpha\sin\theta + \beta\cos\theta\right)\boldsymbol{e}_2
\end{aligned}$$

Ponendo $\alpha = 1$, $\beta = 0$ e $\theta = \dfrac{\pi}{4}$, l'espressione sopra si riduce a:

$$\hat{\mathbf{e}}' = \frac{1}{\sqrt{2}}\left(\boldsymbol{e}_1 - \boldsymbol{e}_2\right)$$

come già visto nel paragrafo precedente.

Abbiamo creato una forma concreta per un operatore di rotazione. Vediamo ora cosa succede all'operatore se agiamo a sinistra con un elemento del duale e a destra lo facciamo agire su un elemento dello spazio vettoriale:

$$\boldsymbol{e}_i^*\left(\boldsymbol{R}\left(\boldsymbol{e}_j\right)\right) = R_{ij}$$

Ad esempio:

$$R_{11} = \boldsymbol{e}_1^*\left(\boldsymbol{R}\left(\boldsymbol{e}_1\right)\right) = \cos\theta$$

In maniera analoga, definiamo gli elementi $R_{12}$, $R_{21}$ e $R_{22}$:

$$R_{12} = \boldsymbol{e}_1^* (\boldsymbol{R} (\boldsymbol{e}_2)) = \sin\theta$$
$$R_{21} = \boldsymbol{e}_2^* (\boldsymbol{R} (\boldsymbol{e}_1)) = -\sin\theta$$
$$R_{22} = \boldsymbol{e}_2^* (\boldsymbol{R} (\boldsymbol{e}_2)) = \cos\theta$$

Come si vede, questi possono essere pensati come gli elementi di una matrice $2 \times 2$:

$$\mathbf{R} = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Questa relazione tra operatori e matrici è alla base del nome **meccanica delle matrici** dato inizialmente da Heisenberg alla meccanica quantistica. Nel resto del testo non sarà necessario eseguire direttamente questi calcoli, ma è importante sapere cosa si intende per rappresentazione di un operatore rispetto agli stati (o vettori) di base.

## 4.19 Operatori unitari

Consideriamo ora un fotone il cui stato sia scritto come:

$$\hat{\mathbf{e}} = \alpha\, \boldsymbol{e}_1 + \beta\, \boldsymbol{e}_2$$

La probabilità che una misura della polarizzazione del fotone dia come risultato $\boldsymbol{e}_1$ (polarizzazione orizzontale) o $\boldsymbol{e}_2$ (polarizzazione verticale) è data dal modulo quadrato dei coefficienti $\alpha$ e $\beta$. Poiché la polarizzazione del fotone ammette solo due stati possibili, in base a quanto visto prima, sappiamo che:

$$|\alpha|^2 + |\beta|^2 = 1$$

Pensiamo ora a un operatore $\boldsymbol{O}$ che agisca sul vettore $\hat{\mathbf{e}}$ trasformandolo come segue:

$$\hat{\mathbf{e}}' = \boldsymbol{O}\, \hat{\mathbf{e}} = \alpha'\, \boldsymbol{e}_1 + \beta'\, \boldsymbol{e}_2$$

Se l'operatore $\boldsymbol{O}$ rappresenta una trasformazione fisica come, ad esempio, una rotazione nello spazio, allora dobbiamo aspettarci che la trasformazione non abbia influito sulla natura fisica del fotone, caratterizzato dai suoi due possibili stati di polarizzazione. Quindi, dopo la trasformazione, anche la somma dei moduli quadrati dei nuovi coefficienti $\alpha'$ e $\beta'$ deve essere pari a 1:

$$\left|\alpha'\right|^2 + \left|\beta'\right|^2 = 1$$

Questo significa che i due stati $\boldsymbol{e}_1$ e $\boldsymbol{e}_2$ costituiscono ancora una base per il fotone. Ci aspettiamo infatti che rimanga invariata la probabilità totale che la polarizzazione del fotone sia verticale od orizzontale, cioè che l'operatore agisca conservando la norma del vettore di stato.

Non tutti gli operatori agiscono in questo modo: gli operatori che godono di questa proprietà sono detti **operatori unitari**.

## 4.20 Trasformazione di base per uno stato

Abbiamo visto che le misure della polarizzazione ammettono due soli possibili risultati: verticale o orizzontale, e che questo ci permette di assumere i vettori $\hat{\mathbf{i}}$ e $\hat{\mathbf{j}}$ come base per rappresentare ogni possibile stato di polarizzazione del fotone.

Da quanto visto nel paragrafo precedente, segue però che ogni altra base ottenuta dalla prima per mezzo di una **trasformazione unitaria** (cioè da un operatore unitario) conserva la probabilità e quindi può essere una base accettabile per descrivere lo stato di un fotone.

La base usata finora è detta **base standard**. Agendo sui vettori della base standard con l'operatore $R$ definito dalla matrice unitaria:

$$\mathbf{R} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix}$$

(si ricordi che $i$ è l'unità immaginaria dei numeri complessi) si ottiene la base data dai vettori:

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

detta **base di polarizzazione circolare**. Questa base è molto importante nell'ottica quantistica, ma in questo testo non ne faremo uso perché non serve ai fini introduttivi che ci siamo proposti.

### 4.20.1 Vettori bra e ket

Nell'introduzione informale, abbiamo introdotto i simboli $|0\rangle$ e $|1\rangle$ per rappresentare lo stato del fotone. Siamo ora in grado di collocarli correttamente nella teoria.

Nella meccanica quantistica, i ket $|\psi\rangle$ sono vettori di uno spazio detto **spazio di Hilbert**, mentre i bra $\langle\psi|$ sono i duali dei vettori (elementi dello spazio duale).

I ket della base standard per la polarizzazione del fotone sono $|0\rangle$ e $|1\rangle$, che corrispondono ai vettori $\hat{\mathbf{i}}$ e $\hat{\mathbf{j}}$ rispettivamente.

È possibile ora derivare la relazione tra la rappresentazione **quantistica**, che usa bra e ket, e quella **ondulatoria** della famosa equazione di Schrödinger, che usa la funzione d'onda $\psi(x)$.

Si supponga che un sistema fisico sia descritto dal ket $|a\rangle$. Lo stesso stato fisico può essere descritto per mezzo della funzione d'onda $\psi(x)$. La relazione tra i due è la seguente:

$$\psi(x) = \langle x|a\rangle$$

dove $|x\rangle$ rappresenta lo stato di posizione, cioè un autostato dell'operatore posizione associato alla coordinata $x$. Spesso si usa anche scrivere $|\psi\rangle$ per indicare un dato stato fisico.

Nel seguito del testo non sarà fatto uso della funzione d'onda di Schrödinger, che è stata introdotta per completezza.

1. Sì
2. No

## 4.21 Osservabili fisiche

Abbiamo visto che, nella teoria quantistica, lo stato di un sistema fisico è rappresentato da un ket, ed è naturale chiedersi come sono rappresentate nella teoria le grandezze fisiche come la quantità di moto, il momento angolare, il campo elettrico, il campo magnetico, ecc.

In questo testo, queste grandezze non verranno usate direttamente, per cui non è necessario entrare nella descrizione dettagliata di questo aspetto della teoria. È interessante però sapere che esse sono rappresentate da operatori detti **osservabili**.

Per essere un'osservabile, un operatore deve essere hermitiano (o autoaggiunto), cioè deve soddisfare la proprietà:

$$\hat{O} = \hat{O}^\dagger$$

dove $\hat{O}^\dagger$ è l'operatore aggiunto (coniugato trasposto) di $\hat{O}$. Questo implica che gli autovalori dell'operatore sono reali, caratteristica fondamentale per le quantità fisiche misurabili.

In questo capitolo si è visto cosa si intende per stato fisico di un sistema quantistico, come è possibile associare un qubit allo stato di un fotone e, cosa molto importante, si è visto cosa si intende per stato entangled. Si è inoltre sviluppato il formalismo necessario per poter leggere e interpretare le espressioni usate nei libri e negli articoli professionali. Con questo bagaglio faticosamente guadagnato, si può ora procedere con i prossimi capitoli per capire, senza compromessi e semplificazioni, i principi dell'informatica quantistica.

## 4.22 Conclusione

In questo capitolo, abbiamo intrapreso un viaggio attraverso i fondamenti matematici della meccanica quantistica, esplorando concetti chiave come gli stati quantistici, il principio di sovrapposizione, gli spazi vettoriali e il prodotto tensoriale. Abbiamo visto come lo stato di un qubit possa essere rappresentato come una combinazione lineare di stati base e come

questa rappresentazione si estenda a sistemi più complessi attraverso il prodotto tensoriale, portando alla possibilità di stati *entangled*.

Abbiamo anche introdotto la notazione bra-ket di Dirac, che fornisce un linguaggio elegante e potente per descrivere gli stati quantistici e le loro trasformazioni. Inoltre, abbiamo discusso il ruolo degli operatori unitari e delle osservabili fisiche, elementi fondamentali per comprendere come gli stati quantistici evolvono e come possono essere misurati.

Siamo consapevoli che la matematica presentata può essere risultata impegnativa e che alcuni concetti possono aver richiesto uno sforzo significativo per essere compresi appieno. Tuttavia, è importante ricordare che l'obiettivo principale era fornire una base concettuale solida per i temi che affronteremo nei capitoli successivi. Anche se alcuni dettagli matematici possono essere sfuggiti, l'essenza dei concetti può essere colta e apprezzata anche senza una comprensione completa del formalismo matematico.

Se hai trovato difficoltà nella parte matematica, non preoccuparti! È perfettamente normale incontrare ostacoli quando si esplorano nuovi campi del sapere, specialmente in un'area complessa come la meccanica quantistica. Il punto cruciale è aver afferrato le idee fondamentali:

- **Stato quantistico**: rappresentato da un vettore (ket) in uno spazio di Hilbert, che può essere una sovrapposizione di stati base.
- **Sovrapposizione**: la capacità di un sistema quantistico di trovarsi simultaneamente in più stati, portando a fenomeni unici come l'interferenza quantistica.
- **Entanglement**: una correlazione profonda tra sistemi quantistici che non può essere spiegata classicamente, in cui lo stato di un sistema non può essere descritto indipendentemente dallo stato dell'altro.
- **Operatori unitari**: trasformazioni che descrivono l'evoluzione temporale di un sistema quantistico, preservando la norma degli stati e quindi la probabilità totale.
- **Osservabili**: grandezze fisiche misurabili associate ad operatori hermitiani, i cui autovalori corrispondono ai possibili risultati delle misurazioni.

Anche senza entrare nei dettagli matematici, comprendere che la meccanica quantistica introduce una nuova visione del mondo, in cui la probabilità e la sovrapposizione giocano un ruolo centrale, è già un grande passo avanti.

Nel prossimo capitolo, ci addentreremo in concetti ancora più affascinanti: l'*entropia quantistica* e la *reversibilità del calcolo quantistico*. L'entropia, in termini semplici, è una misura dell'incertezza o del disordine di un sistema. In informazione quantistica, l'entropia ci aiuta a quantificare la quantità di informazione che possiamo ottenere da un sistema quantistico e come questa informazione cambia durante i processi quantistici.

La reversibilità del calcolo quantistico è un altro concetto chiave. A differenza dei processi classici, molti dei quali sono irreversibili (come la cancellazione di un bit, che aumenta l'entropia del sistema), le operazioni quantistiche descritte da operatori unitari sono

intrinsecamente reversibili. Questo significa che, in linea di principio, possiamo invertire l'evoluzione di un sistema quantistico per tornare allo stato iniziale. Questa proprietà ha profonde implicazioni per la computazione quantistica, rendendo possibile la progettazione di algoritmi più efficienti e riducendo il consumo energetico associato al calcolo.

Comprendere l'entropia e la reversibilità ci permetterà di esplorare temi come:

- **Algoritmi quantistici**: come il calcolo quantistico può risolvere problemi in modo più efficiente rispetto ai metodi classici.
- **Criptografia quantistica**: l'utilizzo di principi quantistici per garantire comunicazioni sicure.
- **Correzione degli errori quantistici**: come mantenere l'informazione quantistica intatta nonostante la presenza di rumore e decoerenza.

Vogliamo incoraggiarti a proseguire con entusiasmo questo viaggio nell'affascinante mondo dell'informatica quantistica. Anche se alcuni concetti possono sembrare astratti o difficili da afferrare, ricorda che ogni grande scoperta è il risultato di curiosità, perseveranza e passione per la conoscenza.

Non scoraggiarti se alcuni dettagli matematici non sono ancora chiari. Molte idee si chiariranno man mano che avanzeremo, e vedrai come i concetti introdotti finora si intrecciano per formare una comprensione più completa del campo. Sentiti libero di rileggere le sezioni precedenti, di porre domande e di approfondire gli argomenti che più ti interessano.

Il mondo quantistico è pieno di meraviglie e sorprese, e siamo appena all'inizio della nostra esplorazione. Nei capitoli successivi, metteremo in pratica i concetti appresi, vedremo come vengono utilizzati nei protocolli di comunicazione quantistica e scopriremo come gli scienziati stanno lavorando per realizzare computer quantistici sempre più potenti.

**Il meglio deve ancora venire!** Preparati a immergerti nei misteri dell'entropia quantistica e a scoprire come la reversibilità del calcolo apra nuove frontiere nell'informatica. Continua con curiosità e mente aperta: il futuro dell'informazione quantistica è nelle tue mani.

## 4.23 Bibliografia ragionata

La meccanica quantistica rappresenta una delle rivoluzioni più profonde nella storia della fisica, ridefinendo i concetti fondamentali di stato fisico, misura e realtà stessa. Per comprendere appieno i principi esposti in questo capitolo, è essenziale fare riferimento a opere chiave che hanno plasmato la nostra comprensione di questa teoria.

Un testo fondamentale è quello di Dirac [1958], *The Principles of Quantum Mechanics*, in cui Paul A. M. Dirac presenta una formulazione elegante e rigorosa della meccanica quantistica. In questo libro, Dirac introduce la notazione bra-ket, fornendo gli strumenti matematici necessari per descrivere gli stati quantistici e le loro trasformazioni. La sua

visione sottolinea l'importanza di accettare la meccanica quantistica non come una semplice estensione della fisica classica, ma come una nuova struttura teorica con i propri principi.

Per comprendere il contesto storico e filosofico in cui la meccanica quantistica è emersa, il lavoro di Jammer [1989], *The Conceptual Development of Quantum Mechanics*, offre un'analisi dettagliata dello sviluppo concettuale della teoria. Jammer esplora le sfide affrontate dai pionieri della meccanica quantistica, come Heisenberg, Schrödinger e Born, mettendo in luce le tensioni tra diverse interpretazioni e la loro influenza sulla formulazione finale della teoria.

La **meccanica ondulatoria** di Erwin Schrödinger è approfondita nel suo lavoro originale Schrödinger [1926], *Quantisierung als Eigenwertproblem*. In questa serie di articoli, Schrödinger introduce l'equazione d'onda che porta il suo nome, offrendo una rappresentazione continua e intuitiva dei fenomeni quantistici. La sua visione contrasta con quella di Heisenberg, che sviluppò la **meccanica matriciale**, un approccio più astratto basato sulle matrici e sugli operatori, come presentato nel suo articolo pionieristico Heisenberg [1925], *Über quantentheoretische Umdeutung kinematischer und mechanischer Beziehungen*.

L'unificazione dei due approcci fu realizzata attraverso il lavoro di Dirac [1927], che mostrò come la meccanica ondulatoria e quella matriciale fossero formalmente equivalenti, gettando le basi per la moderna formulazione della meccanica quantistica. Questo consolidamento è cruciale per comprendere come diversi formalismi matematici possano descrivere gli stessi fenomeni fisici.

Per una trattazione più moderna e didattica dei principi della meccanica quantistica, il libro di Sakurai and Napolitano [2017], *Modern Quantum Mechanics*, offre una panoramica approfondita, combinando rigore matematico e intuizione fisica. Sakurai esplora concetti avanzati come l'entanglement, gli operatori unitari e gli spazi di Hilbert, che sono fondamentali per l'informatica quantistica.

Il principio di indeterminazione di Heisenberg, uno dei pilastri della meccanica quantistica, è discusso in dettaglio nel lavoro di Heisenberg [1927], *Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik*. Questo principio stabilisce i limiti fondamentali nella precisione con cui possiamo conoscere simultaneamente coppie di grandezze fisiche come posizione e quantità di moto, influenzando profondamente la nostra comprensione della realtà a livello microscopico.

Per comprendere le implicazioni filosofiche e interpretative della meccanica quantistica, l'articolo di Bohr [1928], *The Quantum Postulate and the Recent Development of Atomic Theory*, è essenziale. Niels Bohr introduce il principio di complementarità e sviluppa l'interpretazione di Copenaghen, che enfatizza il ruolo fondamentale dell'osservatore e della misura nei fenomeni quantistici.

Infine, per un approfondimento matematico sugli spazi vettoriali, gli operatori e il formalismo necessario per la meccanica quantistica, il testo di Reed and Simon [1980], *Methods of Modern Mathematical Physics*, fornisce una trattazione rigorosa degli strumenti matematici utilizzati nella teoria. Questo libro è particolarmente utile per comprendere

concetti avanzati come i prodotti tensoriali, gli spazi di Hilbert e gli operatori lineari, che sono stati discussi in questo capitolo.

Queste opere costituiscono una base solida per chiunque desideri approfondire i principi della meccanica quantistica, offrendo sia il contesto storico che gli strumenti matematici necessari. Comprendere questi fondamenti è cruciale per apprezzare le innovazioni e le applicazioni dell'informatica quantistica che verranno esplorate nei capitoli successivi.

## 4.24 Applicazioni pratiche con QuantumSim

Per consolidare i concetti introdotti in questo capitolo, utilizziamo **QuantumSim** per simulare l'azione di un operatore unitario su uno stato quantistico e osservare come questo trasforma lo stato rappresentato sulla Sfera di Bloch.

### 4.24.1 Simulazione di una rotazione quantistica

Consideriamo un operatore unitario che rappresenta una rotazione di $\theta$ attorno all'asse Y sulla Sfera di Bloch. L'obiettivo è mostrare come un qubit nello stato iniziale $|0\rangle$ venga trasformato e misurato dopo l'applicazione della rotazione.

```c
#include "quantum_sim.h"
#include <stdio.h>
#include <math.h>

#define M_PI acos(-1.0)

int circuit(void) {
    // Inizializza lo stato quantistico con 1 qubit
    QubitState* state = initializeState(1);

    // Stato iniziale |0>
    initializeStateTo(state, 0);
    printf("Stato iniziale:\n");
    printState(state);

    // Rotazione attorno all'asse Y
    double theta = M_PI / 4; // Angolo di rotazione (45 gradi)
    double complex rotation[2][2] = {
        {cos(theta / 2), -sin(theta / 2)},
        {sin(theta / 2), cos(theta / 2)}
    };
    applySingleQubitGate(state, 0, rotation);
    printf("Stato dopo la rotazione:\n");
    printState(state);

    // Misurazione
    MeasurementResult result = measure(state, 0);
    printf("Risultato della misurazione: %d\n", result.result);

```

```
30      // Libera la memoria allocata
31      freeState(state);
32      return 0;
33  }
```

Listing 4.1: circuit4.c

**Spiegazione:**

- Lo stato iniziale del qubit è $|0\rangle$.
- L'operatore unitario applica una rotazione di $\theta$ attorno all'asse Y, modificando lo stato del qubit sulla Sfera di Bloch.
- La misura collassa lo stato in $|0\rangle$ o $|1\rangle$, con probabilità dipendenti dall'angolo $\theta$.

Compilando ed eseguendo il codice, è possibile osservare come la rotazione influisce sullo stato quantistico e sui risultati della misura.

## Transizione verso il Capitolo 5

Nel **Capitolo 5**, approfondiremo i gate quantistici, strumenti fondamentali per manipolare e controllare i qubits. Analizzeremo il loro funzionamento, l'implementazione nei circuiti quantistici e il loro impatto sulla computazione quantistica. Preparatevi a scoprire il cuore operativo dei computer quantistici.

# Capitolo 5

# I calcoli dei computer consumano energia?

## 5.1 Introduzione

Nel mondo moderno, i computer sono diventati strumenti indispensabili in quasi ogni aspetto della vita quotidiana, dalla comunicazione alla ricerca scientifica, dalla gestione aziendale all'intrattenimento. Tuttavia, con l'aumento esponenziale della potenza di calcolo e della complessità delle operazioni svolte, emerge una domanda cruciale: *I calcoli dei computer consumano energia?* La risposta a questa domanda non solo ha implicazioni pratiche per l'efficienza energetica e la sostenibilità ambientale, ma tocca anche i fondamenti teorici della computazione e della fisica.

Tra i primi a interessarsi a questo problema vi sono stati i fisici Bennett [1973], Fredkin and Toffoli [1982], Toffoli [1980], Feynman [1982] e Deutsch [1985]. All'inizio degli anni '80, Bennett [1973] suggerì che un computer potesse essere pensato come una macchina che trasforma **energia libera** in calore con lo scopo di compiere un *lavoro matematico*. Questa definizione, sebbene possa sembrare inusuale, diventa più intuitiva quando si considera il computer dal punto di vista meccanico.

In effetti, i computer sono meccanismi costruiti usando componenti elettronici che dissipano calore sia per elaborare i dati sia per mantenerli nella memoria volatile. Quindi, da questa prospettiva, la definizione di Bennett [1973] non è poi così peregrina come potrebbe sembrare a prima vista.

Alcuni decenni prima, von Neumann [1966] aveva condotto i primi studi sugli elaboratori elettronici e sugli automi cellulari. Uno dei suoi argomenti di ricerca era stato proprio il calcolo dell'energia associata alla computazione e, con maggior precisione, si era interessato al calcolo dell'aumento dell'**entropia** durante la computazione.

Contemporaneamente, Shannon [1948] introdusse il concetto di **entropia dell'informazione**, applicando la statistica al concetto di informazione e gettando le basi della teoria dell'informazione. Questo concetto si è rivelato fondamentale per comprendere co-

me l'entropia, una misura del disordine o dell'incertezza, si relazioni con la quantità di informazione elaborata dai computer.

In questo capitolo, esploreremo in profondità la relazione tra calcolo computazionale ed energia consumata, analizzando come i principi della termodinamica e della teoria dell'informazione influenzino l'efficienza energetica dei computer. Approfondiremo concetti chiave come l'**entropia dell'informazione**, l'**entropia algoritmica** e la **computazione reversibile**, evidenziando come questi elementi siano fondamentali per lo sviluppo di sistemi di calcolo più sostenibili e performanti.

In particolare, discuteremo:

- **Entropia e informazione**: Comprendere come l'entropia, una misura del disordine o dell'incertezza, si relazioni con la quantità di informazione elaborata dai computer.
- **Principio di Landauer**: Esplorare il limite termodinamico inferiore alla dissipazione di energia associata alla cancellazione di un bit di informazione.
- **Computazione reversibile**: Analizzare come operazioni logiche reversibili possano ridurre il consumo energetico, minimizzando la dissipazione di calore.
- **Circuiti reversibili**: Studiare l'implementazione pratica di circuiti logici reversibili, come le porte Toffoli, e le loro implicazioni per l'efficienza energetica.
- **Implicazioni per l'informatica quantistica**: Collegare i concetti di computazione reversibile e entropia algoritmica con le potenzialità offerte dall'informatica quantistica, inclusi algoritmi più efficienti e riduzione del consumo energetico.

Attraverso questo percorso, mireremo a fornire una visione completa di come i calcoli dei computer consumino energia, le leggi fisiche che ne regolano il funzionamento e le innovazioni tecnologiche che promettono di rendere l'informatica del futuro più efficiente e sostenibile. Comprendere questi principi non solo è essenziale per lo sviluppo di nuove tecnologie, ma anche per affrontare le sfide globali legate al consumo energetico e all'impatto ambientale dei sistemi di calcolo.

## 5.2 Entropia

L'entropia $S$ è una grandezza fisica molto importante nella descrizione dei sistemi fisici che sono coinvolti in processi di trasformazione, come ad esempio una pentola d'acqua che si sta scaldando fino all'ebollizione per cuocere la pasta, o un caminetto in cui la legna arde trasformando l'energia chimica in calore. Questa grandezza è altrettanto importante per descrivere in modo completo anche i sistemi che producono, trasformano e trasmettono *informazione*.

### 5.2.1 Definizione in fisica dell'entropia

Il concetto di entropia è stato introdotto da Carnot e Clausius per quantificare la tendenza naturale dei sistemi fisici ad evolvere verso una precisa direzione anziché un'altra. Per esempio, lasciando cadere una goccia di inchiostro in un bicchiere d'acqua, ci si aspetta che l'inchiostro si diffonda in modo disordinato (direzione attesa) anziché formando delle figure geometriche regolari (direzione alternativa).

Nei termini di temperatura ($T$) e calore ($Q$), l'entropia è definita come:

$$S = \sum_i \frac{dQ_i}{T_i}$$

dove $T_i$ è la temperatura assoluta e $dQ_i$ è il calore scambiato in una trasformazione composta da un insieme discreto di passaggi, ciascuno indicato con l'indice $i$.

Al fine di comprendere le motivazioni dell'informatica quantistica, è molto istruttiva la formulazione data da Boltzmann. Quest'ultima definizione di entropia è basata sulla statistica: molto rigorosa e soprattutto generalizzabile anche ad ambiti diversi dalla fisica.

L'entropia $S$ può essere scritta in termini statistici come:

$$S = k \sum_j p_j \ln\left(\frac{1}{p_j}\right)$$

dove $p_j$ è la probabilità che il sistema si trovi nello stato $j$-esimo e $k$ è la costante di Boltzmann.

**Box: entropia nel gioco dei dadi**

Si consideri di avere a disposizione sei dadi e di disporli in modo tale che la somma dei loro numeri dia 36. Perché si verifichi questa condizione, è necessario che tutti i dadi mostrino la faccia con il numero 6 rivolta verso l'alto. Per ogni dado, esiste un solo caso su sei corrispondente alla faccia 6, quindi esiste un'unica disposizione di dadi che porta il sistema dei sei dadi a dare come somma 36.

Si consideri ora la situazione in cui la somma dei dadi dia come risultato il numero 21.

Usando la definizione di entropia data poco sopra, risulta immediato verificare che l'entropia per la disposizione dei sei dadi tutti risultanti nel numero 6 è bassa, mentre la disposizione risultante nei dadi che danno come somma 21 ha un'alta entropia.

Questa analisi qualitativa ci è sufficiente per intuire che un sistema che si trovi in uno stato a bassa entropia scivolerà più facilmente verso uno stato ad entropia più alta, piuttosto che viceversa.

**Esperimento mentale**

Per comprendere ancora più a fondo, si può fare un semplice esperimento mentale. Si immagini che i dadi siano contenuti in un vaso e che si trovino nello stato iniziale, ordinato, con tutte le facce rivolte verso il 6, in modo che la somma dia 36. Questa situazione corrisponde a un minimo di entropia.

Ora si scuota il vaso in modo che i dadi cambino il loro valore casualmente. Quando ogni dado si sarà fermato, si sommino i valori delle sei facce. Con ogni probabilità si otterrà un valore diverso da 36. Se si ripete l'esperimento molte volte, la media dei valori dopo lo scuotimento del vaso convergerà verso 21.

Questo semplice esperimento ci fornisce un'idea intuitiva di un principio molto importante: in natura, i sistemi tendono ad assumere configurazioni con entropia crescente.

**Entropia e reversibilità**

Sempre continuando con lo stesso esperimento, vediamo che il concetto di entropia è legato anche al concetto di irreversibilità di una trasformazione.

Scelta una faccia da uno a sei, la probabilità che lanciando sei dadi si ottengano sei facce uguali a quella scelta è la stessa per ogni numero. Per esempio, ottenere sei facce con il numero 6 ha la stessa probabilità di ottenere sei facce con il 5, il 4, e così via.

Immaginiamo ora di disporre i dadi allineati in modo che la faccia superiore sia il numero 6 e la faccia frontale sia il 4.

Dallo stato iniziale, in cui tutti i dadi hanno valore 6, incliniamo lentamente il vaso, in maniera che tutti i dadi rotolino nello stesso modo e si posizionino tutti sulla faccia del numero 4. È un esercizio difficile ma, almeno mentalmente, possiamo pensare di riuscire ad eseguirlo.

Ora che tutti i dadi sono nello stato 4, possiamo eseguire il movimento inverso e riportare i dadi sul valore 6: abbiamo eseguito una trasformazione **reversibile**, caratterizzata dal non aver aumentato l'entropia totale del sistema.

L'argomentazione usata ha solo un valore intuitivo; la relazione esatta tra l'entropia e la reversibilità deve essere studiata su un testo di termodinamica dedicato, come ad esempio *Termodinamica* di Enrico Fermi.

Ai fini della comprensione del resto del testo, l'idea intuitiva che abbiamo delineato è sufficiente.

## 5.3 Termodinamica della computazione

In questo paragrafo vengono introdotti due concetti importantissimi nella *teoria dell'informazione* sia classica che quantistica: l'entropia dell'informazione e l'entropia algoritmica. Concetti necessari per capire l'idea di computazione reversibile ed irreversibile, la prima delle quali è alla base dell'informatica quantistica.

### 5.3.1 Entropia dell'informazione

Contemporaneo di von Neumann, il matematico Claude Shannon introdusse il concetto di **entropia dell'informazione** applicando la statistica al concetto di informazione (Shannon, 1948).

In modo molto diretto egli definì lo schema della **teoria dell'informazione** in cinque blocchi uniti da un canale, detto canale dell'informazione. Da un lato si ha la sorgente del-

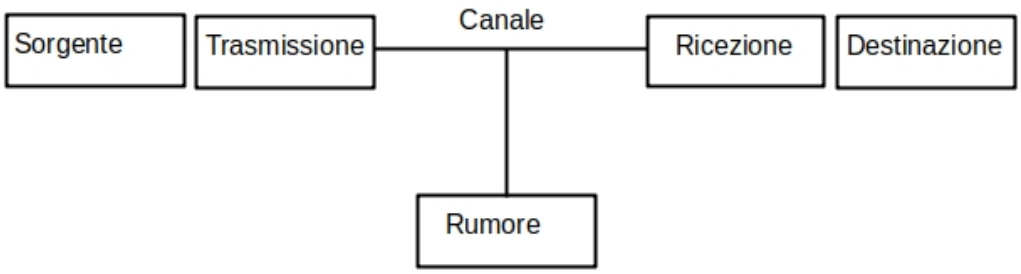



Figura 5.1: Schema del sistema di comunicazione proposto da C. Shannon.

l'informazione e il sistema di trasmissione, mentre dall'altro si trova il sistema di ricezione e la destinazione. Tra i due blocchi si trova la sorgente di *rumore* che può intervenire nella comunicazione alterando l'informazione.

L'informazione viene scambiata per mezzo di messaggi che sono semplicemente delle sequenze di simboli, per esempio i valori 0 e 1.

Partendo dall'idea che per ogni trasmissione l'informazione è trasportata da un messaggio e che ogni messaggio è una sequenza di simboli, egli definì $p_i$ la frequenza con cui appare il simbolo $i$-esimo nel messaggio e mostrò che la funzione $H$ definita come segue:

$$\mathbf{H} = \sum p_i log_2(\frac{1}{p_i}) = -\sum p_i log_2(p_i)$$

rappresentava l'**incertezza** contenuta nel messaggio. (Nota che la scelta della base 2 per il logaritmo, non è vincolante nella teoria.)

### 5.3.2 Un'applicazione della teoria dell'informazione di Shannon

La teoria dell'informazione e il concetto di **incertezza** o **arbitrarietà** contenuta in un messaggio, è piuttosto complessa e merita uno studio dedicato. Ai fini della comprensione di ciò che segue, dobbiamo però notare quanto la definizione di $H$ ricalchi quella data per l'entropia $S$.

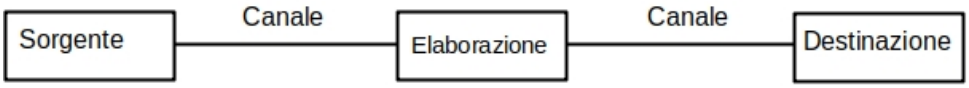



Figura 5.2: Sistema di comunicazione con un elemento di elaborazione dell'informazione tra sorgente e destinatario.

Consideriamo un circuito logico con due variabili di ingresso $a, b$ e due di uscita $a', b'$ composto da una porta AND e una OR definito dalla seguente tabella di verità:

| a | b | a' | b' |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 |

Consideriamo ora una versione modificata dello schema a cinque blocchi visto sopra, in cui tra **sorgente** e **destinazione** introduciamo un blocco di **elaborazione**.

Per inquadrare la situazione secondo la teoria appena illustrata, dobbiamo considerare come simboli della trasmissione non i singoli bit, ma le coppie di bit. Quindi i simboli possibili per ogni messaggio trasmesso dalla sorgente sono:

$\{(1,1),(1,0),(0,1),(0,0)\}$

Nel caso in cui la sorgente abbia arbitrio totale sui simboli da inviare, essi hanno tutti probabilità $p = \frac{1}{4}$, quindi, usando la formulazione vista sopra per l'entropia, calcoliamo che l'**incertezza** dei messaggi trasmessi è 2 .

I possibili messaggi ricevuti, sono invece quelli derivabili dalle due colonne $a', b'$ della tabella e, come si nota, si riducono a soli tre distinti tra loro:

$\{(1,1),(0,1),(0,1),(0,0)\}$

Il risultato dell'elaborazione è quindi quello di aver ridotto il numero di simboli possibili che possono essere ricevuti e di aver aumentato la probabilità (o frequenza) di uno di essi, cioè la coppia $(0,1)$.

È naturale chiedersi se l'elaborazione abbia avuto un effetto sulla quantità $H$ definita poco prima.

Se calcoliamo $H$ tenendo conto di quanto appena visto, vediamo che essa scende dal valore 2 al valore $\frac{3}{2}$. Inoltre notiamo un'altra cosa: alla variazione della funzione $H$ è associato ad un altro concetto importante: il messaggio elaborato è **irreversibile**, cioè da esso non è più possibile risalire al messaggio trasmesso, perché per il simbolo ricevuto $(1,0)$ esistono due diversi possibili simboli trasmessi.

La funzione $H$ è quindi un analogo dell'entropia statistica vista sopra, e per questa analogia, pare che von Neumann abbia suggerito a Shannon di chiamarla entropia dell'informazione. Sembra inoltre che von Neumann abbia corroborato il suo suggerimento

dicendo che visto che in pochi capivano il significato dell'entropia fisica, Shannon non avrebbe avuto critiche presentando il suo lavoro.

Originariamente Shannon aveva sviluppato e applicato la funzione $H$ sopratutto ai problemi di trasmissione dell'informazione disturbati da sorgenti di rumore piuttosto che a trasmissioni trasformate da unità di elaborazione come nell'esempio presentato sopra. Tale argomento viene invece ripreso e sviluppato da Bennet che propone una sintesi del concetto di entropia algoritmica.

## 5.4 Termodinamica della computazione

In questo paragrafo vengono introdotti due concetti importantissimi nella *teoria dell'informazione* sia classica che quantistica: l'entropia dell'informazione e l'entropia algoritmica. Questi concetti sono necessari per capire l'idea di computazione reversibile e irreversibile, la prima delle quali è alla base dell'informatica quantistica.

### 5.4.1 Entropia dell'informazione

Contemporaneo di von Neumann, il matematico Claude Shannon introdusse il concetto di **entropia dell'informazione** applicando la statistica al concetto di informazione Shannon [1948]. In modo molto diretto, egli definì lo schema della **teoria dell'informazione** in cinque blocchi uniti da un canale, detto canale dell'informazione. Da un lato si ha la **sorgente** dell'informazione e il **trasmettitore**, mentre dall'altro si trovano il **ricevitore** e la **destinazione**. Tra i due blocchi si trova la **sorgente di rumore** che può intervenire nella comunicazione alterando l'informazione.

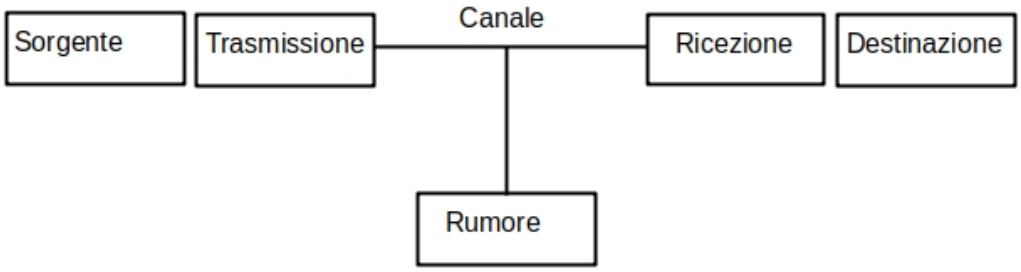



Figura 5.3: Schema del sistema di comunicazione proposto da C. Shannon.

L'informazione viene scambiata per mezzo di messaggi che sono semplicemente delle sequenze di simboli, per esempio i valori 0 e 1.

Partendo dall'idea che per ogni trasmissione l'informazione è trasportata da un messaggio e che ogni messaggio è una sequenza di simboli, egli definì $p_i$ come la frequenza con cui appare il simbolo $i$-esimo nel messaggio e mostrò che la funzione $H$ definita come segue:

$$H = \sum_i p_i \log_2 \left(\frac{1}{p_i}\right) = -\sum_i p_i \log_2 p_i$$

rappresenta l'**incertezza** contenuta nel messaggio. (Nota che la scelta della base 2 per il logaritmo non è vincolante nella teoria.)

### 5.4.2 Un'applicazione della teoria dell'informazione di Shannon

La teoria dell'informazione e il concetto di **incertezza** o **arbitrarietà** contenuta in un messaggio è piuttosto complessa e merita uno studio dedicato. Ai fini della comprensione di ciò che segue, dobbiamo però notare quanto la definizione di $H$ ricalchi quella data per l'entropia $S$.

Consideriamo un circuito logico con due variabili di ingresso $a$, $b$ e due di uscita $a'$, $b'$, composto da una porta AND e una porta OR, definito dalla seguente tabella di verità:

| $a$ | $b$ | $a'$ | $b'$ |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 |

Consideriamo ora una versione modificata dello schema a cinque blocchi visto sopra, in cui tra la **sorgente** e la **destinazione** introduciamo un blocco di **elaborazione**.

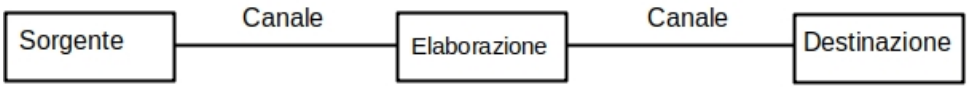



Figura 5.4: Sistema di comunicazione con un elemento di elaborazione dell'informazione tra sorgente e destinatario.

Per inquadrare la situazione secondo la teoria appena illustrata, dobbiamo considerare come simboli della trasmissione non i singoli bit, ma le coppie di bit. Quindi, i simboli possibili per ogni messaggio trasmesso dalla sorgente sono:

$$\{(1,1),\ (1,0),\ (0,1),\ (0,0)\}$$

Nel caso in cui la sorgente abbia arbitrio totale sui simboli da inviare, essi hanno tutti probabilità $p = \frac{1}{4}$. Usando la formulazione vista sopra per l'entropia, calcoliamo che l'**incertezza** dei messaggi trasmessi è:

$$H_{\text{trasmesso}} = -\sum_{i=1}^{4} p_i \log_2 p_i = -4\left(\frac{1}{4}\log_2 \frac{1}{4}\right) = 2$$

I possibili messaggi ricevuti sono invece quelli derivabili dalle due colonne $a'$, $b'$ della tabella, e, come si nota, si riducono a soli tre distinti tra loro:

$$\{(1,1),\ (0,1),\ (0,0)\}$$

Il risultato dell'elaborazione è quindi quello di aver ridotto il numero di simboli possibili che possono essere ricevuti e di aver aumentato la probabilità (o frequenza) di alcuni di essi. Calcoliamo le nuove probabilità:

- Il simbolo $(1,1)$ si verifica solo quando $(a,b) = (1,1)$, quindi ha probabilità $p_{(1,1)} = \frac{1}{4}$. - Il simbolo $(0,1)$ si verifica sia quando $(a,b) = (1,0)$ sia quando $(a,b) = (0,1)$, quindi ha probabilità $p_{(0,1)} = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$. - Il simbolo $(0,0)$ si verifica solo quando $(a,b) = (0,0)$, quindi ha probabilità $p_{(0,0)} = \frac{1}{4}$.

Calcoliamo ora l'entropia dei messaggi ricevuti:

$$\begin{aligned} H_{\text{ricevuto}} &= -\left(p_{(1,1)} \log_2 p_{(1,1)} + p_{(0,1)} \log_2 p_{(0,1)} + p_{(0,0)} \log_2 p_{(0,0)}\right) \\ &= -\left(\frac{1}{4}\log_2 \frac{1}{4} + \frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{4}\log_2 \frac{1}{4}\right) \\ &= -\left(\frac{1}{4} \times (-2) + \frac{1}{2} \times (-1) + \frac{1}{4} \times (-2)\right) \\ &= -\left(-\frac{1}{2} - \frac{1}{2} - \frac{1}{2}\right) = \frac{3}{2} \end{aligned}$$

Quindi, l'entropia $H$ scende dal valore 2 al valore $\frac{3}{2}$ dopo l'elaborazione.

Notiamo un'altra cosa: l'elaborazione è **irreversibile**, cioè da un messaggio ricevuto non è sempre possibile risalire univocamente al messaggio trasmesso. Ad esempio, per il simbolo ricevuto $(0,1)$ esistono due diversi possibili simboli trasmessi: $(1,0)$ e $(0,1)$. Questo implica una perdita di informazione.

La funzione $H$ è quindi analoga all'entropia statistica vista sopra, e per questa analogia pare che von Neumann abbia suggerito a Shannon di chiamarla entropia dell'informazione. Sembra inoltre che von Neumann abbia corroborato il suo suggerimento dicendo che, visto che pochi capivano il significato dell'entropia fisica, Shannon non avrebbe avuto critiche presentando il suo lavoro.

Originariamente, Shannon aveva sviluppato e applicato la funzione $H$ soprattutto ai problemi di trasmissione dell'informazione disturbati da sorgenti di rumore, piuttosto che a trasmissioni trasformate da unità di elaborazione come nell'esempio presentato sopra. Tale argomento viene invece ripreso e sviluppato da Bennett, che propone una sintesi del concetto di entropia algoritmica.

### 5.4.3 Entropia algoritmica

Vediamo ora di sviluppare il concetto di entropia dell'informazione di Shannon verso quello di entropia algoritmica, seguendo la formalizzazione del problema come proposta da **Bennett**.

Egli definisce lo scopo di una computazione come quello di produrre una stringa di *output*

$x$ per mezzo dell'esecuzione di un programma.
Se questo dovesse sembrare strano, si pensi che qualsiasi risultato otteniamo da uno strumento informatico (digitale) è sempre una stringa di bit o una sua successiva *trasduzione* in un diverso segnale. Per esempio, se ascoltiamo della musica da un dispositivo informatico digitale, il suono che arriva alle nostre orecchie è il risultato della conversione di un segnale digitale in uno elettrico analogico e successivamente in un *segnale* acustico (onda di pressione).
Per continuare il ragionamento, definiamo la lunghezza di una stringa come il numero di **bit** da cui è composta. Definiamo poi l'**entropia algoritmica** come il numero minimo di **bit** per configurare un programma che produca l'output $x$.
Per afferrare il significato della grandezza appena introdotta, si provi ad immaginare di voler produrre delle stringhe casuali di bit. In questo caso, il programma non può sfruttare nessuna regola per produrre l'output $x$ in quanto, se le stringhe sono casuali, la successione di 1 e 0 non segue nessuna logica. Per configurare il programma serviranno quindi tanti bit quanti sono quelli che compongono la stringa $x$. Questo è il caso di **massima entropia**.
Adesso, si provi invece a considerare un programma che deve produrre la divisione intera per due di un dato input $a$, cioè $x = \frac{a}{2}$.
Dal punto di vista computazionale, per eseguire questo calcolo basta spostare tutti i bit dell'input $a$ a destra di una posizione; quindi, in linea di principio, il programma può essere configurato usando un unico bit di informazione, perché l'operazione di spostamento a destra deve essere applicata a tutti i bit nello stesso modo. Questo è un caso di **minima entropia**.
Per chiarire ancora di più le idee, si consideri di voler ottenere che la stringa $x$ sia ottenuta dalla divisione per 2 dell'input $a$ nel caso $a$ sia pari, mentre sia il risultato del prodotto per 2 dell'input $a$ nel caso $a$ sia dispari. Semplificando molto il ragionamento, vediamo che sono necessari due bit per configurare il programma che esegue questa elaborazione: il primo per stabilire se $a$ è pari o dispari (in pratica controllando lo stato del bit meno significativo) e il secondo per traslare a destra o a sinistra la stringa $a$.
Probabilmente, a qualcuno non sfuggirà la stretta relazione esistente tra l'entropia algoritmica e la **complessità di Kolmogorov**.

### Entropia di un sistema

Prima di tirare le somme del ragionamento che abbiamo iniziato sull'entropia legata alla computazione, è necessario fare una piccola digressione sull'entropia di un sistema fisico.

La scienza che descrive la trasformazione del calore in lavoro e viceversa è nota con il nome di termodinamica. La termodinamica si basa su tre principi fondamentali e può essere interpretata come una conseguenza statistica della teoria cinetica dei gas.
Nella termodinamica vi sono grandezze macroscopiche, come la temperatura, che sono il risultato del valore medio di grandezze microscopiche come l'energia cinetica delle particelle costituenti il gas. Tali grandezze macroscopiche hanno quindi una *controparte* microscopi-

ca.
L'**entropia**, invece, è una grandezza termodinamica macroscopica, presente sia nella descrizione classica che in quella statistica. Tuttavia, a differenza di altre grandezze, l'entropia non ha una controparte microscopica diretta: non esiste quindi l'entropia di una particella singola, né una grandezza fisica della particella la cui media fornisca il valore dell'entropia del sistema.

Ora che abbiamo apprezzato questa caratteristica dei sistemi fisici, torniamo a considerare l'**entropia algoritmica** e vediamo che questa è l'analogo *microscopico* dell'entropia termodinamica *macroscopica*. In pratica, si può calcolare l'entropia termodinamica se è nota l'entropia algoritmica di un sistema.
Consideriamo un sistema termodinamico il cui comportamento macroscopico sia descrivibile in *modo conciso* Bennett [1973], cioè attraverso delle equazioni del moto macroscopiche, quindi senza bisogno di seguire le componenti del sistema ad una ad una. A questo proposito, Bennett dice esplicitamente:

> "A macrostate is concisely describable if it is determined by equations of motion and boundary conditions, describable in a small number of bits..."

Per un tale sistema, Bennett afferma:

> "...its statistical entropy is nearly equal to the ensemble average of microstates' algorithmic entropy."

In altri termini, Bennett ci dice che per un sistema non del tutto *casuale*, quindi che abbia un certo grado di macro-determinismo, il grado di *casualità* è dovuto all'**entropia algoritmica** delle singole componenti del sistema stesso.
Questo implica che per trasformare la quantità $k_B T \ln 2$ di **calore** in **lavoro** è necessario aumentare di un bit l'entropia algoritmica di un sistema fisico, e corrispondentemente, per diminuire di un bit l'entropia del sistema è necessario fornire un lavoro pari a $k_B T \ln 2$.

### Il principio di Landauer

A questo punto, è fondamentale collegare quanto discusso al **principio di Landauer**. Nel 1961, Rolf Landauer formulò un principio fondamentale che stabilisce una relazione diretta tra informazione ed entropia fisica Landauer [1961]. Il **principio di Landauer** afferma che la cancellazione o la perdita di un bit di informazione in un sistema computazionale comporta necessariamente una dissipazione di energia sotto forma di calore, pari ad almeno:

$$E_{\min} = k_B T \ln 2$$

dove $k_B$ è la costante di Boltzmann e $T$ è la temperatura assoluta del sistema.

Questo principio pone un limite termodinamico inferiore al consumo energetico dei processi computazionali. Ogni volta che un bit di informazione viene cancellato o sovrascritto

in modo irreversibile, l'entropia totale dell'universo aumenta di almeno $k_B \ln 2$, e l'energia corrispondente deve essere dissipata sotto forma di calore.

### Connessione tra Bennett e Landauer

Il risultato finale dell'argomentazione di Bennett, secondo cui per modificare il valore di un bit è necessario spendere almeno un'energia pari a $k_B T \ln 2$, che viene persa in calore, è una generalizzazione del principio di Landauer. Mentre Landauer si concentra sulla dissipazione di energia associata alla cancellazione irreversibile di informazione, Bennett estende questa idea mostrando che la **computazione irreversibile** è intrinsecamente legata all'aumento di entropia e alla dissipazione di energia.

I computer digitali tradizionali sono costituiti da porte logiche che elaborano dati in forma binaria in modo tipicamente irreversibile. Ad esempio, le porte logiche AND e OR perdono informazione sugli stati di ingresso: conoscendo solo l'uscita, non è possibile risalire univocamente agli ingressi. Durante la computazione, ogni porta modifica il valore del proprio bit di output assegnandogli uno tra i valori 0 e 1, indipendentemente dal valore precedente del bit stesso. Quindi, per ogni bit assegnato durante una computazione irreversibile, si ha una **riduzione** di $\ln 2$ unità di entropia nell'apparato di calcolo, che corrisponde a un aumento di entropia nell'ambiente circostante, pari a $k_B \ln 2$, sotto forma di calore dissipato.

Bennett ha mostrato che è possibile progettare computazioni in modo **reversibile**, utilizzando porte logiche reversibili come la **porta di Toffoli**. In una computazione reversibile, non vi è perdita di informazione, e quindi, in linea di principio, non vi è dissipazione minima di energia dovuta al limite di Landauer. Questo apre la possibilità di realizzare computer che consumano meno energia, avvicinandosi al limite termodinamico inferiore.

### Implicazioni per l'informatica quantistica

La computazione quantistica è intrinsecamente **reversibile**, poiché l'evoluzione di un sistema quantistico isolato è descritta da operatori unitari, che sono invertibili. Questo significa che, in linea di principio, i computer quantistici possono eseguire calcoli senza dissipare l'energia minima prevista dal principio di Landauer per le operazioni irreversibili.

Inoltre, grazie ai fenomeni di **sovrapposizione** e **entanglement**, i computer quantistici possono elaborare grandi quantità di informazioni in parallelo, potenzialmente riducendo il numero totale di operazioni necessarie e, di conseguenza, il consumo energetico complessivo.

### Conclusione

Collegando il lavoro di Bennett al principio di Landauer, comprendiamo che la dissipazione di energia nei processi computazionali è strettamente legata alla perdita di informazione e all'aumento di entropia. La computazione irreversibile comporta necessariamente una

dissipazione minima di energia sotto forma di calore, mentre la computazione reversibile offre una via per superare questo limite.

L'informatica quantistica, essendo basata su operazioni reversibili e sfruttando le peculiarità della meccanica quantistica, rappresenta una promettente direzione per realizzare sistemi di calcolo più efficienti dal punto di vista energetico, avvicinandosi al limite termodinamico inferiore imposto dal principio di Landauer.

## 5.5 Termodinamica della computazione reversibile

La valutazione presentata nel paragrafo precedente dell'energia *sprecata* ad ogni assegnamento di un bit si basa sull'assunzione tacita che i processi elettronici usati per scrivere i bit sull'output delle porte logiche siano processi **irreversibili**.

Tra il 1980 e il 1982, furono elaborati due modelli di calcolo alternativi alla computazione ordinaria basati su principi di **calcolo reversibile**: il computer balistico e il computer browniano, che mostrarono esempi concreti, ma non pratici, di computazione reversibile. Nel prossimo paragrafo viene brevemente illustrato solo il computer balistico, il cui principio di funzionamento è più vicino alla meccanica che regola la computazione quantistica, mentre l'approfondimento del computer browniano viene lasciato all'iniziativa del lettore perché, per quanto assolutamente meritevole di approfondimento, ci porterebbe in una direzione diversa da quella voluta.

### 5.5.1 Il computer balistico reversibile

Il computer balistico fu presentato da **Fredkin e Toffoli**, che illustrarono un *esperimento concettuale* per un sistema di calcolo che non dissipa energia, cioè produce *computazione* senza aumentare l'entropia totale del sistema.

Il computer balistico prevede un sistema di input che presenta un numero $n$ di aperture circolari (porte) ed un sistema di output analogo con lo stesso numero di porte.

I **bit** del sistema sono rappresentati da sfere metalliche. Per presentare una stringa di input si preparano un numero $m$ di sfere corrispondenti agli $m$ bit di valore 1 nella stringa di input.

Le sfere vengono inserite con una certa velocità all'interno del computer, dove si trovano solo delle barriere metalliche contro le quali le sfere possono rimbalzare variando (teoricamente) solo la direzione della velocità ma non l'intensità.

Dopo un certo numero di collisioni all'interno dell'elaboratore, le sfere emergono dalle porte di output in una certa disposizione. Le porte da cui emergono le sfere corrispondono ai bit 1 dell'output, mentre quelle da cui non emerge alcuna sfera rappresentano i bit 0.

In condizioni ideali, sia gli urti tra le sfere stesse che quelli tra le sfere e le barriere sono elastici e quindi non trasformano energia cinetica in calore. Quindi, questo tipo di computazione è reversibile.

### 5.5.2 Componenti circuitali per la computazione reversibile

In questo paragrafo vediamo che, in linea di principio, è possibile costruire un **computer reversibile** basato sul concetto comune di porte logiche.

Il computer balistico visto prima, infatti, è senz'altro affascinante, ma difficilmente può rivelarsi una soluzione pratica e realistica per eseguire computazioni di interesse pratico. Per cui vedremo che è possibile costruire una macchina reversibile partendo dai componenti che sono usati concretamente nella realizzazione di normali computer *irreversibili*.

Ogni circuito logico può essere implementato anche usando due sole porte logiche, ad esempio il NOT e la AND.

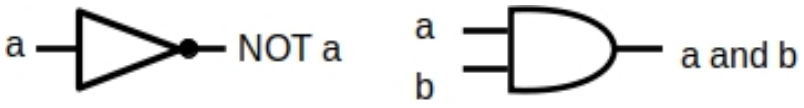



Figura 5.5: Simboli usati per le porte logiche NOT e AND.

Oltre a queste due porte, è di utilità pratica introdurre anche gli elementi FANOUT ed EXCHANGE (vedi figura sotto) che si rivelano indispensabili nella progettazione pratica dei circuiti, dove le variabili di input e output devono essere realizzate concretamente come cavi o linee di conduzione elettrica.

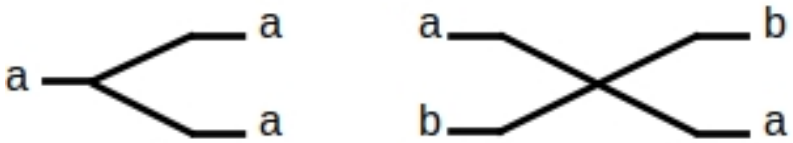



Figura 5.6: Simboli del FANOUT e dell'EXCHANGE.

Partendo da questi quattro elementi circuitali vediamo come realizzarne le *controparti* reversibili e quindi come collegarle tra loro per realizzare dei circuiti logici. La proprietà che accomuna tutte le porte logiche che sono presentate di seguito è che il numero di variabili (indipendenti) di input è uguale al numero di variabili (dipendenti) di output. Senza questa condizione non è possibile ottenere una computazione reversibile.

### 5.5.3 Porta NOT reversibile

La porta NOT produce in output la negazione del segnale di input. Se un bit $a$ viene presentato al suo ingresso, il bit NOT $a$ viene prodotto in uscita dalla porta. La porta NOT viene classicamente rappresentata con un triangolo, ma, seguendo la tradizione di **Feynman**, per evidenziare la natura *simmetrica* della porta NOT, la rappresentiamo con una X sopra un filo (cavo) conduttore con il quale si indica la linea di trasmissione dei bit.

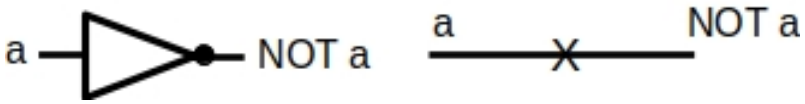



Figura 5.7: Simbolo classico e quantistico della porta NOT.

### 5.5.4 Porta CONTROLLED NOT

Dopo il NOT ci si poteva aspettare di continuare con la porta reversibile AND, ma per farlo c'è bisogno prima di un nuovo elemento circuitale che diverrà molto importante e comune nel proseguimento dello studio della computazione quantistica: il **CONTROLLED NOT** (CNOT). Questa è una porta reversibile e quindi presenta lo stesso numero di ingressi e di uscite che indichiamo rispettivamente con le coppie $a, b$ e $a', b'$.

La tabella di verità delle variabili è rappresentata qui sotto:

| $a$ | $b$ | $a'$ | $b'$ |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 |

Il principio del CNOT è il seguente: il bit $a$ ha il ruolo di *controllore*, mentre il bit $b$ è il vero input. Se il bit $a$ vale 0, allora il bit $b$ viene trasmesso lungo la linea così com'è, quindi $b' = b$. Se invece il bit $a$ vale 1, allora il bit $b$ viene negato. Il CNOT è rappresentato con il simbolo seguente:

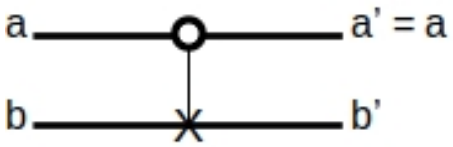



Figura 5.8: Simbolo della porta Controlled NOT (CNOT).

### 5.5.5 FANOUT

La porta CNOT può essere usata per produrre il FANOUT (sdoppiamento di un segnale di ingresso). Se si pone il bit di ingresso $b = 0$, dalla tavola di verità della CNOT si vede che $b' = a$ e $a' = a$, come mostrato nella figura seguente:

### 5.5.6 EXCHANGE

La porta CONTROLLED NOT può essere usata per produrre l'EXCHANGE (incrocio o scambio di segnali).

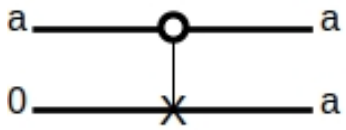



Figura 5.9: FANOUT realizzato con una porta CNOT.

Lo scambio dei bit si ottiene ponendo tre CONTROLLED NOT in serie tra loro, in modo che la linea su cui è presente il bit di controllo risulti alternata tra un CNOT e il successivo. In termini pratici, l'EXCHANGE si ottiene con tre CNOT in serie, di cui quella centrale ha il bit di controllo invertito rispetto ai due CNOT laterali, come mostrato nella figura seguente:

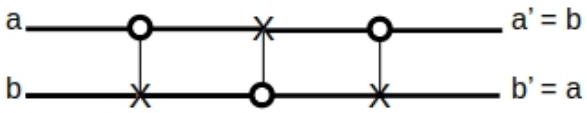



Figura 5.10: EXCHANGE realizzato con porte CNOT.

### 5.5.7 Porta AND reversibile

La porta AND **reversibile** può essere costruita usando un elemento chiamato **porta Toffoli**, o **CONTROLLED CONTROLLED NOT** (CCNOT). Questa porta è fondamentale sia nella computazione classica reversibile che in quella quantistica.

Il funzionamento della porta CCNOT è il seguente: se entrambi i bit di controllo $a$ e $b$ sono settati a 1, allora il bit $c$ di ingresso viene negato, in modo da avere $c' = \text{NOT } c$; altrimenti, il bit $c$ viene trasmesso inalterato, quindi $c' = c$.

La porta AND si ottiene quindi impostando a 0 il bit $c$.

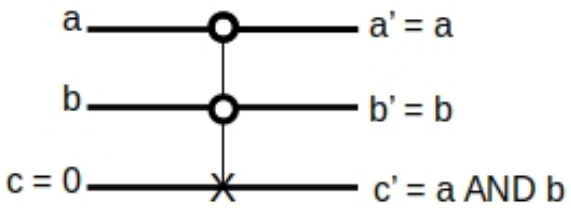



Figura 5.11: Circuito AND reversibile realizzato con una porta CCNOT (porta Toffoli).

La porta CCNOT è reversibile, infatti, guardando la sua tabella di verità:

| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | $c$ | 1 | 0 | $c$ |
| 0 | 1 | $c$ | 0 | 1 | $c$ |
| 0 | 0 | $c$ | 0 | 0 | $c$ |

Si nota che per ogni combinazione di input esiste una sola combinazione di output e viceversa, garantendo così la reversibilità.

#### Il ruolo della porta Toffoli nella computazione quantistica

La **porta Toffoli** è di fondamentale importanza nella computazione quantistica perché è universale per la computazione classica reversibile: qualsiasi funzione booleana può essere realizzata usando solo porte Toffoli. Inoltre, può essere implementata quantisticamente, permettendo di costruire circuiti quantistici che simulano comportamenti classici.

Nella computazione quantistica, la porta Toffoli viene utilizzata per costruire algoritmi che richiedono operazioni controllate su più qubit. La capacità di realizzare operazioni condizionate su più qubit è essenziale per molti algoritmi quantistici, come l'algoritmo di Shor per la fattorizzazione di numeri interi.

### 5.5.8 Circuiti reversibili

Nei paragrafi precedenti sono stati introdotti dei nuovi elementi circuitali per implementare dei circuiti logici. Questi nuovi elementi possono essere usati al posto delle porte logiche tradizionali e producono dei circuiti reversibili.

### 5.5.9 Circuito half-adder

L'addizione tra due bit ($x$ e $y$) vale 0 con resto di 1 se entrambi i bit sono 1, 1 con il resto di 0 se solo uno dei due bit vale 1, ed infine vale 0 con il resto di 0 se entrambi valgono 0.

L'implementazione classica e **irreversibile** di un circuito *half-adder* prevede ad esempio l'uso di una porta AND ed una porta XOR.

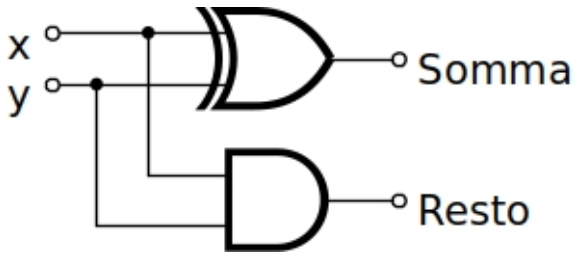



Figura 5.12: Circuito half-adder irreversibile, realizzato con porte logiche. Immagine tratta da *Introduzione alle reti neurali* degli stessi autori.

**Implementazione reversibile del circuito half-adder**

La sua versione reversibile può essere realizzata usando tre linee di ingresso, una porta CNOT e una porta Toffoli (CCNOT), come segue:

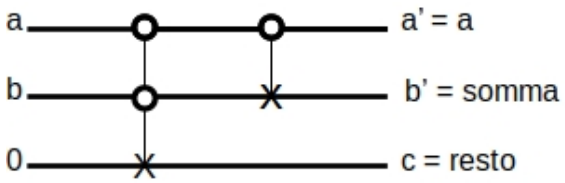



Figura 5.13: Circuito half-adder reversibile realizzato con porte CNOT e CCNOT.

La tabella di verità, costruita usando le regole viste prima:

| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 |

mostra chiaramente che il circuito realizzato:

- implementa correttamente la semi-somma binaria (*half-adder*);
- è reversibile, perché per ogni configurazione di output esiste una sola configurazione di input.

### 5.5.10 Circuito full-adder

L'addizione completa tra due bit prevede che si tenga conto di un eventuale bit di riporto dalla somma del binario precedente $(x, y, r)$.

L'implementazione classica e **irreversibile** di un circuito *full-adder* prevede ad esempio l'uso di due porte AND, due porte XOR ed una porta OR:

**Implementazione reversibile del circuito full-adder**

La sua versione reversibile può essere realizzata usando quattro linee di ingresso, due porte Toffoli (CCNOT) e due porte CNOT, come segue:

Per calcolare la tabella di verità si presti attenzione al fatto che la variabile $d'$ viene modificata due volte lungo il suo canale.

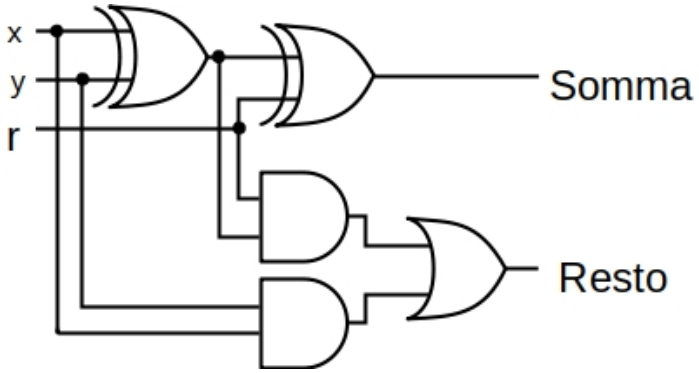



Figura 5.14: Circuito full-adder irreversibile.

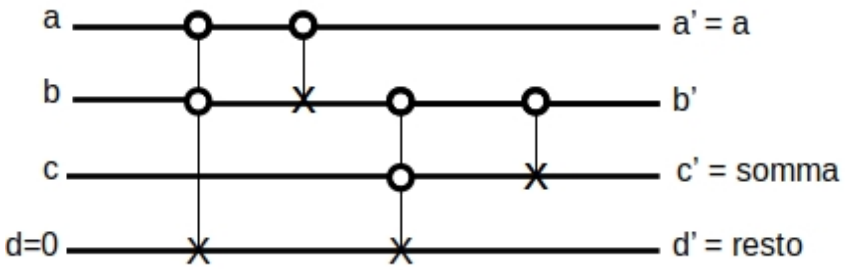



Figura 5.15: Circuito full-adder reversibile nella versione presentata da R. Feynman (1982).

| $a$ | $b$ | $c$ | $d$ | $a'$ | $b'$ | $c'$ | $d'$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

I valori delle variabili $c'$ e $d'$, che corrispondono alla somma e al riporto dell'addizione tra $a$, $b$ e $c$, mostrano che il circuito realizzato:

- implementa correttamente la somma binaria completa (*full-adder*);
- è reversibile, perché per ogni configurazione di output esiste una sola configurazione di input.

### 5.5.11 Importanza della computazione reversibile per minimizzare la dissipazione di energia

La computazione reversibile è fondamentale per minimizzare la dissipazione di energia nei processi computazionali. Come stabilito dal **principio di Landauer**, ogni volta che un bit di informazione viene cancellato in modo irreversibile, si ha una dissipazione minima di energia pari a $k_B T \ln 2$, dove $k_B$ è la costante di Boltzmann e $T$ è la temperatura assoluta del sistema.

Nella computazione irreversibile, come avviene nei computer classici, le operazioni logiche standard (come AND, OR, XOR) perdono informazione sugli stati di ingresso, portando a un aumento dell'entropia e alla dissipazione di energia sotto forma di calore.

La computazione reversibile, invece, preserva l'informazione, permettendo di ricostruire gli stati di ingresso a partire dagli stati di uscita. Questo implica che, in linea di principio, è possibile eseguire calcoli senza dissipare l'energia minima prevista dal principio di Landauer. Tuttavia, nella pratica, la realizzazione di circuiti completamente reversibili presenta sfide significative, ma offre una strada promettente per ridurre il consumo energetico dei computer.

#### Connessione con il principio di Landauer

La riduzione della dissipazione di energia attraverso la computazione reversibile è direttamente collegata al principio di Landauer. Poiché la computazione reversibile evita la cancellazione irreversibile di bit, si può, in teoria, evitare la dissipazione di energia minima associata a tale cancellazione. Questo concetto è fondamentale nella ricerca di computer più efficienti dal punto di vista energetico.

### 5.5.12 Esempi pratici di circuiti reversibili e sfide nella loro implementazione fisica

Nonostante i vantaggi teorici della computazione reversibile, la sua implementazione pratica presenta diverse sfide. I circuiti reversibili richiedono componenti che siano in grado di conservare l'informazione senza dissipare energia, il che è difficile da realizzare con la tecnologia elettronica attuale basata su semiconduttori.

Alcuni esempi di circuiti reversibili pratici includono:

- **Circuiti a logica adiabatica**: questi circuiti utilizzano transizioni energetiche lente per ridurre la dissipazione di energia, seguendo processi quasi reversibili. Tuttavia, richiedono una gestione complessa dell'energia e non eliminano completamente le perdite.

- **Computazione con automi cellulari reversibili**: modelli computazionali come l'automa di **Fredkin** mostrano come sia possibile realizzare computazioni reversibili, ma la loro implementazione fisica su larga scala è ancora una sfida.

- **Computazione quantistica**: i computer quantistici sfruttano qubit che evolvono secondo operazioni unitari, intrinsecamente reversibili. Questo rende la computazione quantistica un candidato ideale per implementare circuiti reversibili.

#### Sfide nella realizzazione fisica

Le principali sfide nella realizzazione fisica di circuiti reversibili sono:

- **Coerenza e decoerenza**: nei sistemi quantistici, mantenere la coerenza dei qubit è fondamentale, ma essi sono suscettibili alla decoerenza causata dall'interazione con l'ambiente, che introduce errori e dissipazione di energia.
- **Controllo preciso**: le operazioni reversibili spesso richiedono un controllo molto preciso dei componenti del circuito, che può essere difficile da ottenere con l'attuale tecnologia.
- **Scalabilità**: costruire circuiti reversibili su larga scala, necessari per applicazioni pratiche, è complesso a causa delle limitazioni tecnologiche e dei costi associati.

Nonostante queste sfide, la ricerca continua a progredire, e nuove tecnologie come l'elettronica molecolare, i semiconduttori avanzati e i sistemi quantistici promettono di avvicinare la computazione reversibile alla realtà pratica.

### 5.5.13 Conclusione

La computazione reversibile rappresenta una frontiera importante nella ricerca di sistemi computazionali più efficienti dal punto di vista energetico. La comprensione e l'applicazione di porte logiche reversibili, come la porta Toffoli, sono fondamentali per sviluppare nuovi paradigmi computazionali che minimizzino la dissipazione di energia, in accordo con il principio di Landauer.

L'integrazione di questi concetti nella computazione quantistica offre prospettive interessanti per il futuro dell'informatica, aprendo la strada a computer più potenti ed efficienti, capaci di risolvere problemi complessi con un minor impatto energetico.

## 5.6 Bibliografia Ragionata

Per approfondire i concetti trattati in questo capitolo, si consiglia la lettura delle seguenti opere, che hanno avuto un ruolo fondamentale nello sviluppo della comprensione della relazione tra calcolo computazionale ed energia.

Shannon [1948], con il suo articolo *A Mathematical Theory of Communication*, ha posto le basi della teoria dell'informazione, introducendo il concetto di **entropia dell'informazione**. Questo lavoro è fondamentale per comprendere come l'entropia, intesa come misura dell'incertezza, sia intrinsecamente legata alla quantità di informazione trasmessa e elaborata nei sistemi di comunicazione e nei computer.

Landauer [1961], nel suo lavoro *Irreversibility and Heat Generation in the Computing Process*, ha formulato il **principio di Landauer**, stabilendo che la cancellazione irreversibile di un bit di informazione comporta una dissipazione minima di energia sotto forma di calore, pari a $k_B T \ln 2$. Questo principio ha profonde implicazioni sulla termodinamica della computazione e sul consumo energetico minimo dei processi computazionali.

Bennett [1973], con l'articolo *Logical Reversibility of Computation*, ha esteso il lavoro di Landauer, mostrando che la computazione può essere realizzata in modo **reversibile**, evitando la dissipazione di energia minima prevista dal principio di Landauer. Bennett introduce il concetto di **computazione reversibile** e descrive come sia possibile progettare circuiti logici reversibili che minimizzano il consumo energetico.

Fredkin and Toffoli [1982] e Toffoli [1980] hanno sviluppato ulteriormente questi concetti, introducendo modelli di calcolo reversibile come il **computer balistico** e la **porta di Toffoli**. Questi lavori sono fondamentali per comprendere come la reversibilità nelle operazioni logiche possa portare a una riduzione significativa della dissipazione energetica nei processi computazionali.

Infine, Feynman [1982] e Deutsch [1985] hanno esplorato le connessioni tra computazione e meccanica quantistica. Feynman [1982], nel suo articolo *Simulating Physics with Computers*, suggerisce che i sistemi quantistici potrebbero essere simulati in modo più efficiente utilizzando computer quantistici. Deutsch [1985] formalizza il concetto di **computer quantistico universale**, mostrando che un computer quantistico potrebbe simulare qualsiasi sistema fisico e che la computazione quantistica è intrinsecamente reversibile.

Queste opere forniscono una base solida per comprendere come i principi della termodinamica e della teoria dell'informazione influenzino il consumo energetico nei processi computazionali e come l'informatica quantistica possa offrire soluzioni per realizzare sistemi di calcolo più efficienti ed energeticamente sostenibili.

## 5.7 Simulazioni pratiche con QuantumSim

Per consolidare i concetti di reversibilità e porte logiche quantistiche introdotti in questo capitolo, utilizziamo **QuantumSim** per implementare e testare alcune delle porte logiche fondamentali e un circuito half-adder reversibile.

### 5.7.1 Simulazione della porta CNOT

La porta CNOT è una componente chiave per implementare la computazione reversibile. La sua tabella di verità indica che il bit di controllo ($a$) rimane invariato, mentre il bit target ($b$) viene negato se $a = 1$.

```
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
```

```
    QubitState* state = initializeState(2); // Due qubit inizializzati a |00>

    // Stato iniziale
    printf("Stato iniziale:\n");
    printState(state);

    // Imposta il primo qubit (controllo) a |1>
    applyX(state, 0);
    printf("Stato dopo X sul primo qubit:\n");
    printState(state);

    // Applica il gate CNOT
    applyCNOT(state, 0, 1);
    printf("Stato dopo CNOT (controllo: qubit 0, target: qubit 1):\n");
    printState(state);

    // Libera la memoria allocata
    freeState(state);
    return 0;
}
```

Listing 5.1: circuit5.1.c

**Spiegazione:**

- Lo stato iniziale è $|00\rangle$.
- Il gate X trasforma il primo qubit in $|1\rangle$, portando lo stato a $|10\rangle$.
- Il gate CNOT utilizza il primo qubit come controllo, invertendo il secondo qubit (target) se il primo vale 1, risultando in $|11\rangle$.

### 5.7.2 Circuito half-adder reversibile

Implementiamo ora un circuito half-adder reversibile, che somma due bit di input ($a$ e $b$) e produce un bit di somma ($S$) e un bit di riporto ($C$).

```
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
    QubitState* state = initializeState(3); // Tre qubit inizializzati a |000>

    // Stato iniziale
    printf("Stato iniziale:\n");
    printState(state);

    // Imposta i bit di input a=1, b=1
    applyX(state, 0); // Imposta il primo qubit (a) a |1>
```

```
    applyX(state, 1); // Imposta il secondo qubit (b) a |1>
    printf("Stato dopo l'inizializzazione degli input:\n");
    printState(state);


    // Applica la porta Toffoli (per il bit di riporto)
    applyToffoli(state, 0, 1, 2);

    // Applica il gate CNOT (per il bit di somma)
    applyCNOT(state, 0, 1);

    printf("Stato dopo l'implementazione del circuito half-adder:\n");
    printState(state);

    // Libera la memoria allocata
    freeState(state);
    return 0;
}
```

Listing 5.2: circuit5.2.c

**Spiegazione:**

- I primi due qubit rappresentano gli input $a$ e $b$.
- Il terzo qubit è inizializzato a 0 e viene utilizzato per calcolare sia il bit di somma ($S$) sia il riporto ($C$).
- Le operazioni CNOT e Toffoli implementano le funzioni logiche del circuito half-adder in modo reversibile.

## Transizione verso il Capitolo 6

Nel **Capitolo 6**, introdurremo concetti avanzati come l'entanglement e il teletrasporto quantistico. Questi fenomeni, basati sull'interazione tra qubits, mostrano il vero potenziale della computazione quantistica rispetto ai sistemi classici. Preparatevi a esplorare i confini dell'informatica quantistica!

# Capitolo 6

# Gate quantistici

## 6.1 Introduzione

I **gate quantistici** rappresentano gli elementi fondamentali nell'elaborazione dell'informazione quantistica, analoghi alle porte logiche nell'informatica classica. Essi operano sui **qubit**, unità base dell'informazione quantistica, manipolando gli stati quantistici attraverso trasformazioni unitarie. Un aspetto cruciale che distingue i gate quantistici dalle loro controparti classiche è la capacità di sfruttare fenomeni tipicamente quantistici come la sovrapposizione e l'entanglement, ampliando significativamente le potenzialità computazionali.

Le **matrici di Pauli** rivestono un ruolo centrale nella meccanica quantistica e, di conseguenza, nell'informatica quantistica. Esse sono utilizzate per definire i gate quantistici elementari, come il gate $X$ (NOT quantistico), il gate $Y$ e il gate $Z$, che corrispondono rispettivamente alle matrici $\sigma_x$, $\sigma_y$ e $\sigma_z$. Questi gate sono essenziali per manipolare lo stato dei qubit e per costruire circuiti quantistici in grado di eseguire algoritmi complessi.

Nel contesto dell'informatica quantistica, comprendere come implementare e combinare questi gate è fondamentale per sviluppare algoritmi quantistici efficaci. Ad esempio, il **gate di Hadamard** ($H$) è utilizzato per creare stati di sovrapposizione, mentre il **gate CNOT** permette di generare entanglement tra qubit, una risorsa chiave per molte applicazioni quantistiche.

In questo capitolo, esploreremo in dettaglio:

- **Il gate NOT come gate quantistico**: Analizzeremo come il gate NOT classico si traduce nel contesto quantistico e come viene rappresentato attraverso la matrice di Pauli $\sigma_x$.
- **Matrici di Pauli**: Approfondiremo le proprietà matematiche delle matrici di Pauli e il loro ruolo nelle trasformazioni quantistiche fondamentali.
- **Gate quantistici elementari**: Esploreremo i gate $X$, $Y$, $Z$ e $H$, comprendendo come agiscono sui qubit e come possono essere implementati fisicamente.

- **Il gate CNOT**: Studieremo il funzionamento del gate CNOT, la sua rappresentazione matematica e il suo ruolo nella creazione di stati entangled.
- **Altri gate a due qubit**: Introdurremo gate più complessi e discuteremo come essi estendano le capacità computazionali dei circuiti quantistici.

Attraverso questo percorso, intendiamo fornire una comprensione solida delle fondamenta matematiche e fisiche dei gate quantistici, evidenziando come essi siano strumenti indispensabili per manipolare e controllare l'informazione quantistica. Comprendere questi concetti è essenziale per chiunque desideri approfondire l'informatica quantistica e contribuire allo sviluppo di tecnologie quantistiche avanzate.

**Nota bene**: nei capitoli introduttivi abbiamo visto come lo stato di polarizzazione di un fotone sia una buona osservabile (vedi paragrafo 3.5) per rappresentare il valore di un qubit. Anche le particelle *fermioniche* (cioè particelle a spin semintero) come l'elettrone, a *spin* uguale a un mezzo, possono essere usate per rappresentare i qubit, così come altri sistemi quantomeccanici che possiedono solo due stati base. In questo testo, considereremo sempre che il valore dei qubit sia associato allo stato di polarizzazione dei fotoni. I concetti illustrati sono indipendenti dalla reale tecnologia usata per ottenere e manipolare lo stato dei qubit.

## 6.2 Il NOT come gate quantistico

Consideriamo il qubit $a$ che abbia valore $|0\rangle$ rispetto alla base standard. Possiamo scriverlo usando la notazione dei ket come segue:

$$a = |0\rangle$$

o equivalentemente come un vettore colonna:

$$\mathbf{a} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Vogliamo vedere come trasformare il valore del qubit $a = |0\rangle$ in $a' = |1\rangle$. Questa trasformazione non è un puro problema matematico, ma deve essere realizzabile a livello sperimentale se vogliamo capire come funziona nella realtà un computer quantistico.

La matematica ci guida nella comprensione dei fenomeni fisici e la fisica è la scienza della realtà; le leggi della fisica descrivono processi che avvengono in natura, quindi se troviamo una trasformazione quantomeccanica che realizza l'operazione matematica voluta, troviamo anche l'operazione fisica da compiere per realizzarla.

Con questa premessa, vediamo che per trasformare il valore di $a$ da $|0\rangle$ a $|1\rangle$, che è un'operazione matematica, è necessario a livello fisico trasformare lo stato di un fotone da $|0\rangle$ a $|1\rangle$, cambiandone la polarizzazione agendo con uno strumento ottico che, come vedremo qui di seguito, dovrà compiere una trasformazione diversa da una semplice rotazione.

Dal punto di vista matematico formale, si tratta di trasformare il vettore

$$\mathbf{a} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

nel vettore

$$\mathbf{a}' = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Detta trasformazione può essere ottenuta operando su $\mathbf{a}$ con la matrice

$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Si noti che il simbolo $X$ usato per indicare la matrice è lo stesso usato nel simbolo del CNOT descritto precedentemente.

Dal punto di vista algebrico, la negazione di un qubit è quindi ottenuta moltiplicando la matrice $X$ per il vettore $\mathbf{a}$ come segue:

$$\mathbf{a}' = \mathbf{X}\mathbf{a}$$

Dal punto di vista fisico, invece, la cosa è più complessa. Non sarà infatti sfuggito che la trasformazione $X$ **non equivale** a una rotazione nel piano. Le rotazioni nel piano $xy$ attorno all'asse $z$, di un angolo $\theta$ misurato in senso antiorario, possono essere rappresentate (vedi paragrafo 3.3) con matrici $2 \times 2$ del tipo

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

che, come si deduce, non possono essere ridotte alla matrice $X$, che ha i due elementi sulla diagonale secondaria dello stesso segno, mentre per ogni angolo $\theta$ gli elementi sulla diagonale secondaria della matrice di rotazione $\mathbf{R}$ hanno segno opposto (tranne quando valgono zero).

Per realizzare fisicamente la trasformazione $X$, si usano strumenti ottici come i *mezzi lambda* (*half-wave plates*), che consistono in cristalli birifrangenti che alterano la polarizzazione dei fotoni.

Il circuito NOT per un singolo qubit è rappresentato graficamente come un quadrato (*box* in inglese) con una X.

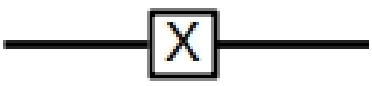



Figura 6.1: Simbolo del gate X (NOT quantistico).

### 6.2.1 Il NOT come trasformazione unitaria

Abbiamo visto che l'operazione NOT di un qubit equivale al prodotto della matrice $X$ per il vettore rappresentante il suo stato di polarizzazione.

La matrice $X$ è una trasformazione unitaria nel senso che $X^\dagger X = I$, dove $X^\dagger$ è la matrice hermitiana coniugata di $X$, e può essere rappresentata in forma operatoriale come segue:

$$X = |0\rangle\langle 1| + |1\rangle\langle 0|$$

La sua azione sui kets di base è data dalle trasformazioni:

$$X|0\rangle = |1\rangle$$

$$X|1\rangle = |0\rangle$$

La negazione di un qubit è quindi un'operazione **quantistica** che agisce nello spazio degli stati del fotone, e questo significa che esiste una azione fisica **reversibile** che trasforma un qubit da $|0\rangle$ a $|1\rangle$ e viceversa, come anticipato nel paragrafo precedente.

Il risultato appena ottenuto ci dice che un'operazione logica come la negazione può essere ottenuta come una trasformazione quantistica reversibile che agisce su un singolo fotone. Quindi possiamo dire che:

> Per modificare il valore di un qubit è necessario trasformare lo stato del fotone che lo rappresenta.

A ben pensarci, un'affermazione equivalente è corretta anche per l'informatica classica, ma in quell'ambito la fisica è più nascosta, mentre emergono problemi più ingegneristici che vengono trattati con più *senso comune* e meno formalismo fisico.

Prima di proseguire, notiamo un aspetto della computazione quantistica di straordinaria importanza **?**, cioè che:

> Ogni calcolo può essere decomposto in trasformazioni che agiscono sul singolo qubit e in trasformazioni che agiscono su due qubit, come il CNOT.

In pratica, dopo aver definito la trasformazione NOT che agisce sul singolo qubit e la trasformazione CNOT che agisce su due qubit, è possibile implementare qualsiasi operazione logica in termini quantistici.

Questo primo risultato evidenzia come la computazione automatica che attualmente è realizzata da circuiti elettronici irreversibili può essere implementata nei termini di trasformazioni quantistiche reversibili. Aver dimostrato che è possibile replicare la computazione *classica* in termini quantistici è ovviamente un passo necessario per dare valore scientifico all'informatica quantistica.

### 6.2.2 Il ruolo delle matrici di Pauli e le rotazioni quantistiche

È importante sottolineare che la matrice $X$ è una delle tre matrici di Pauli, che sono fondamentali nella meccanica quantistica. Le matrici di Pauli sono:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Queste matrici rappresentano le componenti degli operatori di spin e generano le rotazioni nello spazio degli stati quantistici. In particolare, il gate $X$ può essere visto come una rotazione di $\pi$ (180 gradi) attorno all'asse $x$ sulla sfera di Bloch.

La rotazione di un qubit attorno a un asse può essere espressa mediante l'operatore unitario:

$$R_{\mathbf{n}}(\theta) = e^{-i\frac{\theta}{2}(\mathbf{n}\cdot\boldsymbol{\sigma})}$$

dove $\mathbf{n}$ è un vettore unitario che indica l'asse di rotazione, $\theta$ è l'angolo di rotazione, e $\boldsymbol{\sigma} = (X, Y, Z)$ è il vettore delle matrici di Pauli.

Nel caso specifico del gate $X$, si ha:

$$X = e^{-i\frac{\pi}{2}X}$$

Quindi, il gate $X$ corrisponde a una rotazione di $\pi$ attorno all'asse $x$.

### 6.2.3 Implementazione fisica del gate X

Per realizzare fisicamente la trasformazione $X$, nel caso di qubit rappresentati dalla polarizzazione dei fotoni, si può utilizzare un **mezzo lambda** (mezza lunghezza d'onda), noto anche come *half-wave plate*, orientato in modo opportuno. Questo dispositivo ottico introduce un ritardo di fase che inverte la polarizzazione del fotone da orizzontale a verticale e viceversa, realizzando così l'operazione di NOT.

In sistemi quantistici basati su spin di elettroni o di nuclei, la trasformazione $X$ può essere implementata mediante impulsi elettromagnetici a radiofrequenza che inducono transizioni tra i livelli di spin, corrispondenti a una rotazione di $\pi$ attorno all'asse $x$.

### 6.2.4 Universalità dei gate quantistici

Il risultato importante da sottolineare è che combinando operazioni che agiscono su singoli qubit (come le rotazioni generate dalle matrici di Pauli) e operazioni che coinvolgono coppie di qubit (come il gate CNOT), è possibile costruire un insieme universale di gate quantistici. Ciò significa che qualsiasi operazione quantistica può essere approssimata arbitrariamente bene utilizzando combinazioni di questi gate.

Questo concetto è fondamentale nella progettazione di circuiti quantistici e nella realizzazione pratica di algoritmi quantistici.

## 6.3 Matrici di Pauli

La matrice $X$ che abbiamo introdotto nel paragrafo precedente è una delle tre matrici di *Pauli*, definite come segue:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Le matrici di Pauli sono matrici **Hermitiane**, cioè soddisfano $\sigma_i^\dagger = \sigma_i$, dove $\sigma_i^\dagger$ è la matrice trasposta coniugata di $\sigma_i$. Inoltre, le matrici di Pauli sono **unitarie** (a meno di una fase globale), poiché soddisfano $\sigma_i^2 = I$, dove $I$ è la matrice identità:

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

Queste proprietà le rendono fondamentali nella meccanica quantistica, in quanto rappresentano gli operatori di spin per particelle di spin 1/2, come gli elettroni.

Da ognuna delle matrici di Pauli, in quanto matrici unitarie, è possibile *teoricamente* ottenere un corrispondente **quantum gate**.

Per passare dalla *teoria* alla *pratica*, bisogna verificare se esiste nella realtà un *setup sperimentale*, cioè un dispositivo concreto, che agisca sulla polarizzazione dei fotoni o su altri sistemi fisici, così come l'operatore agisce sui kets.

A questa domanda fornirono una risposta di carattere generale Reck, Zeilinger, Bernstein e Bertani, che con una lettera pubblicata sulla rivista *Physical Review Letters* presentarono un procedimento concreto per realizzare in laboratorio una trasformazione ottica da un sistema di input di $N$ stati in un sistema di output di $N$ stati usando solo *beam splitters*, *phase shifters* e *specchi* Reck et al. [1994]. Testualmente scrivono:

> "Per quel che ci risulta, questa è la prima volta che viene presentata una prova pratica che ad ogni operatore unitario discreto può essere associato un esperimento nella realtà."

Da questo, sappiamo che la *matematica* che stiamo scrivendo ha una corrispondenza nella realtà, e abbiamo quindi la stessa fiducia che ha un progettista elettronico nel creare funzioni logiche: sa che esiste un hardware che può implementarle.

### 6.3.1 Gate $Y$ e $Z$

Il gate $Y$ è costruito sulla matrice $\sigma_y$, ma è più pratico definirlo come $Y = -i\sigma_y$, cioè:

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

Il gate $Z$ corrisponde esattamente alla matrice $\sigma_z$:

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

I due gate vengono rappresentati graficamente all'interno di un box quadrato, analogamente al gate $X$.

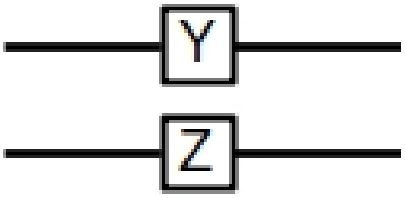



Figura 6.2: Rappresentazione grafica dei gate $Y$ e $Z$.

In termini operatoriali, i gate possono essere scritti come:

$$Y = -i\left(|0\rangle\langle 1| - |1\rangle\langle 0|\right)$$

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1|$$

Vediamo come agiscono questi operatori sugli stati di base e su un qubit generale $|\psi\rangle = a|0\rangle + b|1\rangle$.

**Azione del gate $Z$:** Il gate $Z$ agisce sugli stati base come:

$$Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle$$

Quindi, l'azione di $Z$ su un qubit generico è:

$$Z|\psi\rangle = Z(a|0\rangle + b|1\rangle) = a|0\rangle - b|1\rangle$$

Il gate $Z$ inverte la fase del componente $|1\rangle$ del qubit, lasciando invariato $|0\rangle$.

**Azione del gate $Y$:** Il gate $Y$ agisce sugli stati base come:

$$Y|0\rangle = i|1\rangle, \quad Y|1\rangle = -i|0\rangle$$

Quindi, l'azione di $Y$ su un qubit generico è:

$$Y|\psi\rangle = Y(a|0\rangle + b|1\rangle) = aY|0\rangle + bY|1\rangle = ai|1\rangle - bi|0\rangle = -ib|0\rangle + ia|1\rangle$$

Se ignoriamo il fattore globale $i$, possiamo scrivere:

$$Y|\psi\rangle = -b|0\rangle + a|1\rangle$$

Il gate $Y$ combina una permutazione dei coefficienti $a$ e $b$ con un cambio di segno e una rotazione di fase. Può essere visto come l'azione combinata dei gate $X$ e $Z$, infatti si ha:

$$Y = iXZ = iZX$$

### 6.3.2 Implementazione fisica dei gate $Y$ e $Z$

Per realizzare fisicamente il gate $Z$, si può utilizzare un **ritardatore di fase** (*phase shifter*) che introduce una differenza di fase di $\pi$ (180 gradi) tra le componenti di polarizzazione del fotone.

Il gate $Y$ può essere implementato mediante una combinazione di elementi ottici che introducono sia un cambiamento di fase che una permutazione degli stati, come un *quarter-wave plate* (quarto d'onda) orientato opportunamente.

## 6.4 Il gate di Hadamard ($H$)

Il gate di Hadamard, rappresentato con il simbolo di una $H$ all'interno di un box quadrato, è definito dalla matrice:

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

o in forma operatoriale:

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


Figura 6.3: Simbolo del gate di Hadamard ($H$).

Il gate di Hadamard trasforma gli stati base in una sovrapposizione di stati:

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

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

Agisce quindi come una rotazione che porta gli stati $|0\rangle$ e $|1\rangle$ in stati sovrapposti con uguale peso, ma differente fase relativa.

### 6.4.1 Proprietà del gate di Hadamard

Il gate $H$ è una trasformazione unitaria e auto-inversa, cioè $H^\dagger = H$ e $H^2 = I$.

Applicando due volte il gate $H$, si ritorna allo stato iniziale:

$$HH|\psi\rangle = I|\psi\rangle = |\psi\rangle$$

### 6.4.2 Implementazione fisica del gate di Hadamard

Per realizzare il gate di Hadamard su qubit fotonici, si possono utilizzare combinazioni di *beam splitter* e *specchi* semiriflettenti che creano sovrapposizioni di percorsi, oppure con cristalli birifrangenti orientati in modo da trasformare le polarizzazioni lineari in circolari e viceversa.

### 6.4.3 Utilizzo del gate di Hadamard

Il gate di Hadamard è fondamentale in molti algoritmi quantistici, poiché permette di creare stati di sovrapposizione necessari per il parallelismo quantistico. Vedremo il suo utilizzo nel **dense coding**, nella **teletrasmissione quantistica** e negli algoritmi di **Deutsch-Jozsa** e di **Grover**.

## 6.5 Il gate CNOT

L'azione del **gate CNOT** (**Controlled NOT**) può essere definita solo se si considera un sistema formato da due qubit. Quando questi sono espressi rispetto alla **base standard**, come definito nel capitolo precedente, la sua azione è di *negare* il secondo qubit (qubit target) se il primo qubit (qubit di controllo) è $|1\rangle$, o lasciarlo inalterato se il primo qubit è $|0\rangle$.

Prima di analizzare questo gate, è necessario formalizzare un insieme di kets di base per rappresentare gli stati di due qubit. Ovviamente, la formalizzazione che segue poggia su quanto visto nel paragrafo 3.3.

### 6.5.1 Base standard per due qubit

La base standard per un sistema formato da due qubit è data dal prodotto tensoriale delle due basi singole, quindi dai quattro ket:

$$|0\rangle \otimes |0\rangle, \quad |0\rangle \otimes |1\rangle, \quad |1\rangle \otimes |0\rangle, \quad |1\rangle \otimes |1\rangle$$

che possono essere scritti più concisamente come segue:

$$|00\rangle, \quad |01\rangle, \quad |10\rangle, \quad |11\rangle$$

In forma vettoriale, i quattro ket di base si rappresentano come segue:

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

quindi come quattro vettori colonna, che risultano comodi quando si passa al calcolo matriciale.

### 6.5.2 Rappresentazione operatoriale e matriciale del gate CNOT

Ora che abbiamo definito un insieme di kets di base per gli stati di due qubit, possiamo vedere come si presenta in forma operatoriale il gate CNOT:

$$\text{CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$$

dove $I$ è la matrice identità $2 \times 2$, e $X$ è il gate NOT definito precedentemente.

Espandendo l'operatore, possiamo scrivere il gate CNOT in termini delle basi:

$$\text{CNOT} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10|$$

La rappresentazione matriciale del gate CNOT, rispetto alla base standard ordinata $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, è:

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

In entrambe le rappresentazioni è immediato verificare la correttezza dell'azione di trasformazione del CNOT sugli elementi della base:

$$\text{CNOT}\,|00\rangle = |00\rangle, \quad \text{CNOT}\,|01\rangle = |01\rangle,$$

$$\text{CNOT}\,|10\rangle = |11\rangle, \quad \text{CNOT}\,|11\rangle = |10\rangle$$

Infatti, considerando a titolo di esempio la trasformazione $\text{CNOT}\,|10\rangle$, vediamo che essa può essere scritta come:

$$\text{CNOT}\,|10\rangle = (|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X)\,|10\rangle$$

Poiché $|10\rangle = |1\rangle \otimes |0\rangle$, abbiamo:

$$\text{CNOT}\,|10\rangle = |1\rangle\langle 1|1\rangle \otimes X|0\rangle = |1\rangle \otimes X|0\rangle = |1\rangle \otimes |1\rangle = |11\rangle$$

Analogamente, possiamo verificare le altre trasformazioni.

### 6.5.3 Azione del CNOT su stati in sovrapposizione

Il gate CNOT è un operatore lineare e la sua azione su stati in sovrapposizione si determina applicando il principio di sovrapposizione della meccanica quantistica.

Se consideriamo un generico ket dato dalla seguente espressione:

$$|\psi\rangle = a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle$$

l'azione del CNOT su di esso è la seguente:

$$\text{CNOT}\,|\psi\rangle = a\,|00\rangle + b\,|01\rangle + c\,|11\rangle + d\,|10\rangle$$

Questo perché il gate CNOT scambia i coefficienti $c$ e $d$ associati agli stati $|10\rangle$ e $|11\rangle$.

È importante notare che il gate CNOT si *comporta* come un NOT condizionato solo per qubit che si trovano in uno degli stati di base definiti prima. Tuttavia, essendo un operatore lineare, la sua azione su stati in sovrapposizione segue le regole della meccanica quantistica, e non può essere descritta semplicemente come una funzione logica classica.

Per fissare bene le idee sull'azione del CNOT, consideriamo il seguente stato di input:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\,(|00\rangle + |10\rangle)$$

Questo rappresenta un sistema di due qubit in cui il qubit di controllo è nello stato $\frac{1}{\sqrt{2}}\,(|0\rangle + |1\rangle)$ e il qubit target è nello stato $|0\rangle$. Lo stato $|\psi\rangle$ dei due qubit presi insieme non rappresenta nessuno dei quattro stati di base singolarmente.

Il CNOT agisce sullo stato $|\psi\rangle$ trasformandolo in $|\psi'\rangle$ come segue:

$$|\psi'\rangle = \text{CNOT}\,|\psi\rangle = \frac{1}{\sqrt{2}}\,(\text{CNOT}\,|00\rangle + \text{CNOT}\,|10\rangle) = \frac{1}{\sqrt{2}}\,(|00\rangle + |11\rangle)$$

Nel risultato ottenuto ci sono due cose molto importanti da notare riguardo a $|\psi'\rangle$:

- $|\psi'\rangle$ non rappresenta nessuno dei quattro stati di base del sistema.
- $|\psi'\rangle$ è uno stato **entangled**.

La prima considerazione riflette il fatto che il CNOT, applicato a uno stato in sovrapposizione, produce un nuovo stato che generalmente sarà anch'esso una sovrapposizione di stati base.

La seconda considerazione è che lo stato $|\psi'\rangle = \frac{1}{\sqrt{2}}\,(|00\rangle + |11\rangle)$ è uno stato di **entanglement**, perché non può essere scritto come prodotto tensoriale di stati dei singoli qubit. Vediamo perché.

Supponiamo per assurdo che esistano coefficienti $\alpha$, $\beta$, $\gamma$, $\delta$ tali che:

$$|\psi'\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle)$$

Espandendo il prodotto tensoriale:

$$|\psi'\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$$

Ma noi sappiamo che:

$$|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right)$$

Confrontando le due espressioni, otteniamo le equazioni:

$$\alpha\gamma = \frac{1}{\sqrt{2}}, \quad \alpha\delta = 0, \quad \beta\gamma = 0, \quad \beta\delta = \frac{1}{\sqrt{2}}$$

Dalle equazioni $\alpha\delta = 0$ e $\beta\gamma = 0$, deduciamo che o $\alpha = 0$ o $\delta = 0$, e che o $\beta = 0$ o $\gamma = 0$. Tuttavia, se $\alpha = 0$, allora $\alpha\gamma = 0$, in contraddizione con $\alpha\gamma = \frac{1}{\sqrt{2}}$. Analogamente per gli altri casi. Questo porta a una contraddizione, quindi lo stato $|\psi'\rangle$ non può essere scritto come prodotto di stati dei singoli qubit.

In conclusione, abbiamo che lo stato $|\psi'\rangle$ ottenuto dalla trasformazione CNOT dello stato $|\psi\rangle$ è un esempio di stato **entangled**. In termini fisici, possiamo dire che:

> Lo stato entangled tra due qubit non può essere considerato come lo stato di due qubit separati e considerati insieme, ma è invece lo stato di un unico sistema che va considerato nella sua totalità.

In questo paragrafo abbiamo visto una realtà pratica in cui si producono gli stati fisici entangled che avevamo presentato nel capitolo 3 solo come risultato matematico.

Nel capitolo 6, vedremo che tali stati sono alla base delle caratteristiche peculiari della computazione quantistica.

## 6.6 Altri gate a due qubit

Nel capitolo 4 abbiamo introdotto il simbolo grafico per rappresentare un gate CNOT all'interno di un circuito. Lo stesso simbolo è usato anche per rappresentare il *quantum gate* CNOT che abbiamo appena introdotto.

Il concetto di *gate controllato* può essere esteso anche ad altre trasformazioni oltre alla trasformazione $X$ vista prima. Si usa la lettera $Q$ per indicare una generica trasformazione quantistica, e un generico gate controllato è rappresentato con il simbolo seguente:

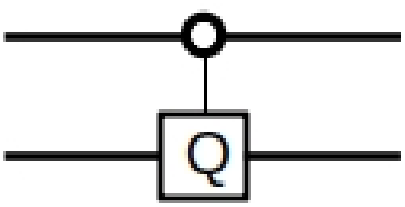



Figura 6.4: Simbolo del generico gate controllato $CQ$.

## 6.7 Bibliografia Ragionata

Per approfondire i temi trattati in questo capitolo, è fondamentale fare riferimento a opere chiave che hanno contribuito in modo significativo allo sviluppo dell'informatica quantistica e alla comprensione delle matrici di Pauli nel contesto della meccanica quantistica.

Nielsen and Chuang [2010], nel loro testo *Quantum Computation and Quantum Information*, offrono una trattazione completa e dettagliata dei fondamenti dell'informatica quantistica. Il libro dedica ampio spazio alla matematica delle matrici di Pauli e alla loro applicazione nella definizione dei gate quantistici. Gli autori esplorano come questi gate siano utilizzati per manipolare i qubit e costruire circuiti quantistici complessi, fornendo esempi ed esercizi che facilitano la comprensione pratica dei concetti teorici.

Mermin [2007], con *Quantum Computer Science: An Introduction*, presentano un'introduzione chiara e accessibile ai principi dell'informatica quantistica. Mermin dedica particolare attenzione alle matrici di Pauli e alla loro interpretazione fisica, utilizzando un approccio pedagogico che rende i concetti complessi più comprensibili per i lettori che si avvicinano per la prima volta all'argomento. Il testo illustra inoltre l'importanza dei gate quantistici elementari e come essi siano utilizzati nella costruzione di algoritmi quantistici.

Nel suo libro *Quantum Theory: Concepts and Methods*, Peres [1995] fornisce una trattazione approfondita della meccanica quantistica, con particolare enfasi sulle basi matematiche e concettuali. Peres esplora in dettaglio le matrici di Pauli, discutendo le loro proprietà algebriche e il loro ruolo nella rappresentazione degli operatori di spin. Questo testo è essenziale per comprendere le fondamenta teoriche che sottendono l'utilizzo delle matrici di Pauli nell'informatica quantistica e nella fisica delle particelle.

Sakurai and Napolitano [2014], con *Modern Quantum Mechanics*, offrono un'analisi avanzata dei concetti fondamentali della meccanica quantistica. Il libro include una discussione approfondita sulle matrici di Pauli, le loro applicazioni nelle trasformazioni di spin e le implicazioni per i sistemi a due livelli, come i qubit. Sakurai fornisce anche una panoramica sulle rotazioni nello spazio degli stati quantistici, collegando le matrici di Pauli alle operazioni di rotazione sulla sfera di Bloch, un concetto chiave per visualizzare le operazioni sui qubit.

Le *Lectures on Physics* di Feynman et al. [2011], in particolare il volume 3, presentano una trattazione classica della meccanica quantistica. Feynman introduce le matrici di Pauli nel contesto della teoria dell'elettrone e dello spin, offrendo intuizioni preziose sulla loro interpretazione fisica. Le sue lezioni sono note per la chiarezza e l'approccio intuitivo, rendendo questo testo una risorsa inestimabile per chi desidera comprendere a fondo i principi della meccanica quantistica e la loro applicazione nell'informatica quantistica.

Infine, l'articolo di Reck et al. [1994], *Experimental realization of any discrete unitary operator*, dimostra sperimentalmente come sia possibile implementare qualsiasi operazione unitaria discreta utilizzando componenti ottici. Questo lavoro è particolarmente rilevante per l'implementazione fisica dei gate quantistici basati sulle matrici di Pauli, mostrando come le trasformazioni teoriche possano essere realizzate concretamente in laboratorio. Gli

autori presentano un procedimento pratico per realizzare in laboratorio una trasformazione ottica da un sistema di input di $N$ stati in un sistema di output di $N$ stati, utilizzando solo beam splitter, phase shifter e specchi.

Queste opere, insieme, forniscono una panoramica completa sia degli aspetti teorici che pratici delle matrici di Pauli e del loro ruolo nell'informatica quantistica. La loro lettura è fortemente consigliata per chiunque desideri approfondire la comprensione dei gate quantistici e delle fondamenta matematiche della computazione quantistica.

## 6.8 Esercizi pratici con QuantumSim

Per esplorare i concetti di sovrapposizione e entanglement, utilizziamo **QuantumSim** per simulare l'applicazione dei gate $H$ e CNOT. Questi esercizi permettono di comprendere meglio le trasformazioni degli stati quantistici e i principi fondamentali della meccanica quantistica.

### 6.8.1 Simulazione del gate Hadamard

Il gate Hadamard crea una sovrapposizione di stati. Lo stato $|0\rangle$ viene trasformato in $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

```c
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
    QubitState* state = initializeState(1); // Inizializza un qubit

    // Stato iniziale |0>
    printf("Stato iniziale:\n");
    printState(state);

    // Applica il gate Hadamard
    applyHadamard(state, 0);
    printf("Stato dopo Hadamard:\n");
    printState(state);

    // Misura
    MeasurementResult result = measure(state, 0);
    printf("Risultato della misurazione: %d\n", result.result);

    // Libera la memoria allocata
    freeState(state);
    return 0;
}
```

Listing 6.1: circuit6.1.c

**Spiegazione:**

- Lo stato iniziale è $|0\rangle$.
- Dopo l'applicazione di $H$, il qubit è in una sovrapposizione equa tra $|0\rangle$ e $|1\rangle$.
- La misura collassa lo stato in $|0\rangle$ o $|1\rangle$, con probabilità del 50% ciascuna.

### 6.8.2 Simulazione di entanglement con il gate CNOT

Il gate CNOT crea uno stato entangled quando applicato a due qubit, con il primo in una sovrapposizione e il secondo inizializzato a $|0\rangle$.

```c
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
    QubitState* state = initializeState(2); // Due qubit inizializzati a |00>

    // Stato iniziale
    printf("Stato iniziale:\n");
    printState(state);

    // Applica il gate Hadamard sul primo qubit
    applyHadamard(state, 0);
    printf("Stato dopo Hadamard sul primo qubit:\n");
    printState(state);

    // Applica il gate CNOT
    applyCNOT(state, 0, 1);
    printf("Stato dopo CNOT (entanglement):\n");
    printState(state);

    // Libera la memoria allocata
    freeState(state);
    return 0;
}
```

Listing 6.2: circuit6.2.c

**Spiegazione:**

- Il primo qubit è inizializzato in $|0\rangle$ e trasformato in $|+\rangle$ con il gate Hadamard.
- Il gate CNOT entangla il primo e il secondo qubit, producendo lo stato $\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$.
- Questo stato non può essere separato in singoli stati dei qubit, evidenziando il fenomeno dell'entanglement.

## Transizione verso il Capitolo 7

Nel **Capitolo 7**, esploreremo applicazioni avanzate dell'entanglement, come il **teletrasporto quantistico** e il **dense coding**. Questi protocolli dimostrano il potenziale unico dell'entanglement per la comunicazione e l'elaborazione quantistica.

# Capitolo 7

# Computazione quantistica

## 7.1 Introduzione

Negli ultimi decenni, la computazione quantistica è emersa come un campo rivoluzionario che promette di superare i limiti dei computer classici. Mentre i computer tradizionali elaborano informazioni in bit che assumono valori di 0 o 1, i computer quantistici utilizzano i **qubit**, che possono esistere in una sovrapposizione di stati grazie alle proprietà della meccanica quantistica. Questo permette di esplorare simultaneamente molteplici percorsi computazionali, aprendo la strada a algoritmi che possono risolvere problemi complessi in tempi significativamente ridotti rispetto alle controparti classiche.

Nei capitoli precedenti, abbiamo esplorato come la computazione classica possa essere ripensata in termini **reversibili**, permettendo una corrispondenza diretta tra i componenti classici reversibili e le trasformazioni quantistiche che agiscono a livello atomico e subatomico. Questa comprensione getta le basi per comprendere le potenzialità uniche offerte dalla computazione quantistica.

In questo capitolo, approfondiremo tre caratteristiche peculiari che distinguono l'informatica quantistica da quella classica:

- **Il teorema di non clonazione dei qubit**: Dimostreremo come, a differenza dei bit classici, i qubit non possano essere copiati in modo perfetto, una proprietà fondamentale che ha implicazioni profonde per la sicurezza e la trasmissione dell'informazione quantistica.
- **Il dense coding**: Esploreremo come sia possibile trasmettere più informazioni rispetto ai limiti classici attraverso l'uso dell'entanglement, permettendo la comunicazione di due bit classici inviando un solo qubit.
- **Il teletrasporto quantistico**: Analizzeremo il protocollo che consente di trasferire lo stato quantistico di un qubit da un luogo a un altro, senza spostare fisicamente il qubit stesso, ma utilizzando l'entanglement e la comunicazione classica.

Queste caratteristiche non solo evidenziano le potenzialità uniche della computazione quantistica, ma sottolineano anche le profonde differenze concettuali rispetto all'informatica classica. Ad esempio, il teorema di non clonazione introduce limitazioni nella manipolazione dell'informazione quantistica, rendendo necessarie nuove strategie per la correzione degli errori e la sicurezza delle comunicazioni. Allo stesso tempo, il dense coding e il teletrasporto quantistico mostrano come l'entanglement possa essere sfruttato per superare i limiti tradizionali di comunicazione e elaborazione dell'informazione.

Approfondendo questi concetti, il capitolo fornirà una comprensione più profonda delle fondamenta teoriche della computazione quantistica e delle sue applicazioni pratiche. Discuteremo anche le implicazioni per la sicurezza informatica e le sfide associate alla realizzazione pratica di questi protocolli. In un'epoca in cui la tecnologia quantistica sta rapidamente avanzando, una solida comprensione di questi principi è essenziale per rimanere al passo con le innovazioni future.

## 7.2 Teorema di non clonazione

Abbiamo ora gli elementi teorici che permettono di capire veramente il fascino e il valore di questo teorema senza doversi accontentare di una spiegazione qualitativa.

Il principio di non clonazione afferma che:

> Non è possibile copiare in modo affidabile lo stato di un sistema quantistico $A$ in un altro sistema quantistico $B$ se lo stato di $A$ non è noto.

Abbiamo visto che le operazioni sui qubit vengono eseguite attraverso **trasformazioni unitarie**. Dimostriamo il teorema mostrando un caso avverso.

Supponiamo (**per assurdo**) che sia possibile copiare lo stato di un sistema sconosciuto in un secondo sistema, e che esista un operatore unitario $U$ per eseguire tale trasformazione.

Sia $|\psi\rangle$ lo stato ignoto del sistema da clonare e $|0\rangle$ lo stato iniziale (fissato) del sistema in cui si desidera clonare $|\psi\rangle$. Allora, si dovrebbe avere:

$$U\left(|\psi\rangle \otimes |0\rangle\right) = |\psi\rangle \otimes |\psi\rangle$$

Supponiamo ora di avere due stati ortogonali $|a\rangle$ e $|b\rangle$, e consideriamo la loro sovrapposizione $|c\rangle = \frac{1}{\sqrt{2}}\left(|a\rangle + |b\rangle\right)$. Per la linearità degli operatori unitari, si dovrebbe avere:

$$U\left(|c\rangle \otimes |0\rangle\right) = U\left(\frac{1}{\sqrt{2}}\left(|a\rangle + |b\rangle\right) \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}\left(U\left(|a\rangle \otimes |0\rangle\right) + U\left(|b\rangle \otimes |0\rangle\right)\right)$$

Applicando l'ipotesi che $U$ cloni gli stati $|a\rangle$ e $|b\rangle$, abbiamo:

$$U\left(|a\rangle \otimes |0\rangle\right) = |a\rangle \otimes |a\rangle \quad \text{e} \quad U\left(|b\rangle \otimes |0\rangle\right) = |b\rangle \otimes |b\rangle$$

Quindi:

$$U\left(|c\rangle \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}\left(|a\rangle \otimes |a\rangle + |b\rangle \otimes |b\rangle\right)$$

D'altra parte, se $U$ clona lo stato $|c\rangle$, dovrebbe valere:

$$\begin{aligned} U\left(|c\rangle \otimes |0\rangle\right) &= |c\rangle \otimes |c\rangle \\ &= \left(\frac{1}{\sqrt{2}}\left(|a\rangle + |b\rangle\right)\right) \otimes \left(\frac{1}{\sqrt{2}}\left(|a\rangle + |b\rangle\right)\right) \\ &= \frac{1}{2}\left(|a\rangle \otimes |a\rangle + |a\rangle \otimes |b\rangle + |b\rangle \otimes |a\rangle + |b\rangle \otimes |b\rangle\right) \end{aligned}$$

Si osserva che le due espressioni sono diverse. In particolare, nel primo caso non compaiono i termini misti $|a\rangle \otimes |b\rangle$ e $|b\rangle \otimes |a\rangle$, mentre nel secondo caso sì.

Questa contraddizione dimostra che non esiste un operatore unitario $U$ che possa clonare uno stato quantistico arbitrario. Quindi, il teorema di non clonazione è dimostrato.

È importante notare che in certi casi particolari potrebbe essere possibile clonare stati specifici, ma poiché esistono stati per i quali non è possibile, una trasformazione unitaria che cloni tutti gli stati non può esistere.

Inoltre, il teorema di non clonazione si applica solo a stati ignoti. Se si conosce lo stato di un sistema, è possibile preparare un altro sistema nello stesso stato quantistico attraverso un processo di preparazione, non di clonazione.

Una nota per chi ha già conoscenza della meccanica quantistica: si faccia attenzione a non mal interpretare quest'ultima affermazione che potrebbe apparire in contraddizione con il **principio di esclusione di Pauli**. Infatti, anche nel caso di qubit fermionici, la copia di uno stato significa preparare un secondo sistema in quello stesso stato, e quindi non è in contraddizione con detto principio, poiché gli stati si riferiscono a particelle distinte.

### 7.2.1 Conseguenze del teorema di non clonazione

A causa del teorema enunciato, nella computazione di qubit e nella loro trasmissione non è possibile applicare le tecniche di rivelazione e correzione degli errori usate nella computazione classica, che spesso si basano sulla duplicazione dell'informazione (ad esempio, la codifica di ripetizione).

La computazione quantistica e la trasmissione di dati quantistici sono soggette a diverse sorgenti di errori: rumore, decoerenza, errori di trasformazione dovuti ai gate, errori di preparazione dei qubit e di misura degli stessi. Sarebbe difficile, per questo motivo, accettare la computazione quantistica se non esistesse una tecnica di correzione degli errori.

Per fortuna, questa è stata individuata e sviluppata a metà degli anni '90, e fa largo uso dell'entanglement e della ridondanza quantistica. La sua descrizione e comprensione va però oltre gli obiettivi di questo testo, dove ci limitiamo a sottolineare l'importanza dell'argomento. Le codifiche di correzione degli errori quantistici, come il codice di Shor o il codice di Steane, permettono di proteggere l'informazione quantistica senza violare il teore-

ma di non clonazione, utilizzando stati entangled di più qubit per distribuire l'informazione in modo protetto.

Il teorema di non clonazione ha ovviamente molta rilevanza dal punto di vista della sicurezza dei dati. Si faccia però attenzione al fatto che il teorema afferma che non sia possibile clonare dei dati con precisione, ma lascia uno spiraglio a chi volesse clonarli accettando un certo grado di imprecisione: un attaccante potrebbe tentare di intercettare e misurare parzialmente lo stato quantistico, introducendo disturbi nel sistema.

Comunque bisogna sempre stare attenti, perché la sicurezza informatica è un tema caldo e attuale anche con le tecnologie quantistiche, e i rischi si possono nascondere dove meno uno se li aspetta, come nei *loopholes quantistici* **?**.

### 7.2.2 I Loopholes quantistici

I cosiddetti **loopholes quantistici** sono potenziali vulnerabilità nei protocolli quantistici che possono essere sfruttate da un avversario per compromettere la sicurezza. Ad esempio, nel campo della crittografia quantistica, sono stati identificati vari loopholes che riguardano imperfezioni nei dispositivi utilizzati, come rivelatori non ideali o sorgenti di fotoni non perfettamente singoli. Questi loopholes possono essere sfruttati per attacchi come l'intercettazione e il reinvio o l'attacco di *blinding*.

Lo studio di tali loopholes è fondamentale per garantire la sicurezza dei protocolli quantistici nella pratica, poiché la sicurezza teorica deve essere accompagnata da implementazioni che tengano conto delle limitazioni reali dei dispositivi.

## 7.3 Dense Coding

Il **dense coding** permette di comunicare due bit (non qubit) di informazione trasmettendo un solo qubit: da qui il nome *dense* (denso). Questa possibilità non esiste nell'informatica e nella meccanica classica, ma non si deve pensare che sia una specie di *magia* quantistica. Infatti, sappiamo che:

> Da ogni qubit è possibile estrarre un solo bit di informazione.

Quindi, come avviene il dense coding? La spiegazione è che il dense coding si basa sulla condivisione *pre-comunicazione* di uno stato **entangled**, dove *pre-comunicazione* significa che, prima della trasmissione, i due soggetti della comunicazione avevano in precedenza condiviso una risorsa comune, analogamente a quello che potrebbe essere la condivisione di un disco su una rete locale.

### 7.3.1 Preparazione dello stato entangled

Lo stato entangled è la risorsa che i due condividono. Può sembrare un concetto astratto, ma, in termini pratici, si tratta di realizzare un dispositivo che prepara un sistema fisico microscopico in uno stato quantistico entangled.

Per esempio, è possibile preparare uno stato entangled dall'emissione, da parte di un atomo di calcio, di due fotoni emessi in *cascata*, cioè uno di seguito all'altro. La cascata di due fotoni si ottiene eccitando i livelli atomici dell'atomo di calcio, che successivamente torna nel suo stato fondamentale passando in successione prima dal livello $6^1P_1$ al livello $4^1S_0$ e successivamente al $4^1P_1$.

Per eccitare i livelli atomici, il calcio viene semplicemente scaldato. I fotoni che vengono emessi in direzioni opposte sono entangled ed hanno entrambi polarizzazione verticale oppure entrambi polarizzazione orizzontale, e possono essere usati per il dense coding **?**.

Supponiamo che la risorsa condivisa sia lo stato entangled di due fotoni come descritto sopra. Possiamo descriverlo come:

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

Si ricordi che la scrittura:

$$|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle$$

equivale alla seguente:

$$|00\rangle + |11\rangle$$

In questo contesto preferiamo la prima alla seconda perché evidenzia il fatto che il sistema fisico è condiviso tra due soggetti; quindi, possiamo pensare al primo ket come associato al soggetto 1 e il secondo al soggetto 2. Oppure, ricordandoci dei signori Fabbri e Magri introdotti nei primi capitoli, possiamo riscrivere lo stato $|\psi\rangle$ specificando il ket di Fabbri con la lettera maiuscola $F$ e il ket di Magri con la $M$:

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(|0_F\rangle \otimes |0_M\rangle + |1_F\rangle \otimes |1_M\rangle\right)$$

### 7.3.2 Codifica nel dense coding

Questa volta il signor Fabbri vuole comunicare un'informazione al signor Magri. Il signor Fabbri, infatti, ha necessità di informare Magri circa la scelta da lui compiuta di uno dei quattro punti cardinali, questo perché Fabbri e Magri gestiscono un servizio di orientamento.

Il signor Fabbri trasforma il proprio fotone della coppia entangled secondo il seguente schema:

- **Nord** $\rightarrow (I \otimes I)\,|\psi\rangle = \frac{1}{\sqrt{2}}\,($
  $|0_F\rangle \otimes |0_M\rangle + |1_F\rangle \otimes |1_M\rangle\ )$
- **Sud** $\rightarrow (X \otimes I)\,|\psi\rangle = \frac{1}{\sqrt{2}}\,($
  $|1_F\rangle \otimes |0_M\rangle + |0_F\rangle \otimes |1_M\rangle\ )$

- **Ovest** $\rightarrow (Z \otimes I)\,|\psi\rangle = \frac{1}{\sqrt{2}}\,($
$$|0_F\rangle \otimes |0_M\rangle - |1_F\rangle \otimes |1_M\rangle\ )$$
- **Est** $\rightarrow (iY \otimes I)\,|\psi\rangle = \frac{1}{\sqrt{2}}\,($
$$|1_F\rangle \otimes |0_M\rangle - |0_F\rangle \otimes |1_M\rangle\ )$$

Le trasformazioni $X$, $Y$, $Z$ e $I$ sono quelle viste nel capitolo precedente. Ricordiamo che il prodotto $U \otimes I$ indica che la trasformazione $U$ è applicata al primo qubit (quello di Fabbri), mentre il secondo è lasciato inalterato, anche perché il secondo qubit non si trova in possesso del signor Fabbri.

Quindi, se il signor Fabbri ha scelto l'**Est** e vuole comunicarlo a Magri, anzitutto trasforma lo stato $|\psi\rangle$ nel ket:

$$|\psi'\rangle = \frac{1}{\sqrt{2}}\,(|1_F\rangle \otimes |0_M\rangle - |0_F\rangle \otimes |1_M\rangle)$$

agendo con l'operatore $iY$ sul suo qubit, mentre il qubit di Magri (che non è a disposizione di Fabbri) rimane invariato perché trasformato per mezzo dell'identità $I$ che non sortisce alcun effetto.

### 7.3.3 Trasmissione e ricezione

Dopo la trasformazione operata sul proprio qubit, il signor Fabbri lo trasmette al signor Magri, il quale non misura subito lo stato del qubit per estrarre qualche informazione, ma trasforma la coppia di qubit ora in suo possesso con due operazioni in successione: prima applica un gate CNOT (con il qubit di Fabbri come controllo e il suo qubit come target) e poi applica l'operatore di Hadamard ($H$) sul qubit di Fabbri.

Dopo tali trasformazioni, procede alla misura dello stato dei due qubit, giungendo a una delle quattro possibili combinazioni dei due qubit:

- $|0_F\rangle \otimes |0_M\rangle$
- $|0_F\rangle \otimes |1_M\rangle$
- $|1_F\rangle \otimes |0_M\rangle$
- $|1_F\rangle \otimes |1_M\rangle$

Il signor Magri, avendo eseguito i calcoli, sa che la prima combinazione corrisponde al **Nord**, la seconda al **Ovest**, la terza al **Sud** e la quarta all'**Est**.

Riassumendo, è possibile comunicare due bit classici di informazione trasmettendo un singolo qubit.

### 7.3.4 Trasformazione e decodifica

Il signor Magri riceve il qubit descritto dal ket $|\psi'\rangle$ e lo trasforma in $|\psi''\rangle$ agendo prima con un gate CNOT e poi con un Hadamard gate sul qubit di Fabbri:

$$|\psi''\rangle = (H \otimes I)\ \mathrm{CNOT}\,|\psi'\rangle$$

Di seguito, esegue la misura dello stato dei due qubit, ottenendo uno dei quattro possibili stati di base. Vediamo nei dettagli il calcolo esplicito dell'azione delle trasformazioni sullo stato $|\psi'\rangle$ per ciascuna delle quattro possibili scelte di Fabbri.

#### Calcoli espliciti

I quattro possibili stati dopo la trasformazione di Fabbri sono:

- **Nord**:
$$|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|0_F\rangle \otimes |0_M\rangle + |1_F\rangle \otimes |1_M\rangle\right)$$
- **Sud**:
$$|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|1_F\rangle \otimes |0_M\rangle + |0_F\rangle \otimes |1_M\rangle\right)$$
- **Ovest**:
$$|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|0_F\rangle \otimes |0_M\rangle - |1_F\rangle \otimes |1_M\rangle\right)$$
- **Est**:
$$|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|1_F\rangle \otimes |0_M\rangle - |0_F\rangle \otimes |1_M\rangle\right)$$

Applichiamo il gate CNOT, dove il qubit di Fabbri è il controllo e quello di Magri è il target. Ricordiamo che il gate CNOT agisce come:

$$\mathrm{CNOT}\,|a_F\rangle \otimes |b_M\rangle = |a_F\rangle \otimes |b_M \oplus a_F\rangle$$

Applicando il CNOT ai quattro stati sopra, otteniamo:

- **Nord**:
$$\mathrm{CNOT}\,|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|0_F\rangle \otimes |0_M\rangle + |1_F\rangle \otimes |0_M\rangle\right)$$
- **Sud**:
$$\mathrm{CNOT}\,|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|1_F\rangle \otimes |1_M\rangle + |0_F\rangle \otimes |1_M\rangle\right)$$
- **Ovest**:
$$\mathrm{CNOT}\,|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|0_F\rangle \otimes |0_M\rangle - |1_F\rangle \otimes |0_M\rangle\right)$$
- **Est**:
$$\mathrm{CNOT}\,|\psi'\rangle = \frac{1}{\sqrt{2}}\left(|1_F\rangle \otimes |1_M\rangle - |0_F\rangle \otimes |1_M\rangle\right)$$

Ora, applichiamo l'operatore $H \otimes I$, dove $H$ agisce sul qubit di Fabbri:

$$H|0_F\rangle = \frac{1}{\sqrt{2}}\left(|0_F\rangle + |1_F\rangle\right), \quad H|1_F\rangle = \frac{1}{\sqrt{2}}\left(|0_F\rangle - |1_F\rangle\right)$$

Applichiamo $H$ ai risultati precedenti:

- **Nord**:

$$|\psi''\rangle = \frac{1}{2}\left((|0_F\rangle + |1_F\rangle) \otimes |0_M\rangle + (|0_F\rangle - |1_F\rangle) \otimes |0_M\rangle\right) = |0_F\rangle \otimes |0_M\rangle$$

- **Sud**:

$$|\psi''\rangle = \frac{1}{2}\left((|0_F\rangle - |1_F\rangle) \otimes |1_M\rangle + (|0_F\rangle + |1_F\rangle) \otimes |1_M\rangle\right) = |1_F\rangle \otimes |0_M\rangle$$

- **Ovest**:

$$|\psi''\rangle = \frac{1}{2}\left((|0_F\rangle + |1_F\rangle) \otimes |0_M\rangle - (|0_F\rangle - |1_F\rangle) \otimes |0_M\rangle\right) = |0_F\rangle \otimes |1_M\rangle$$

- **Est**:

$$|\psi''\rangle = \frac{1}{2}\left((|0_F\rangle - |1_F\rangle) \otimes |1_M\rangle - (|0_F\rangle + |1_F\rangle) \otimes |1_M\rangle\right) = -|1_F\rangle \otimes |1_M\rangle$$

Notiamo che il segno negativo nello stato finale per l'**Est** non ha conseguenze misurabili, poiché gli stati differiscono per una fase globale non osservabile.

### 7.3.5 Conclusione

Grazie al dense coding, è possibile trasmettere due bit classici di informazione inviando un solo qubit, sfruttando uno stato entangled precondiviso. Questo risultato mostra come l'entanglement possa essere utilizzato per aumentare la capacità di comunicazione rispetto ai limiti classici.

## 7.4 Teletrasporto

Il **teletrasporto quantistico** di uno stato quantico è forse la più suggestiva delle novità portate dalla meccanica e dalla computazione quantistica. Bisogna però stare attenti e non lasciarsi prendere troppo dalla fantasia. Nel teletrasporto, ad essere teletrasportata è *un'informazione* e **non** la materia o l'energia in sé. In pratica, quello che si ha è che l'informazione codificata in un qubit non è più accessibile in un luogo mentre diventa accessibile in un altro.

### 7.4.1 Setup sperimentale

Riprendiamo i signori Fabbri e Magri introdotti nei paragrafi precedenti. Il signor Fabbri ha un qubit rappresentato dal ket $|\phi\rangle$ che vuole trasmettere al signor Magri attraverso un canale di informazione classico, cioè inviandogli due bit. Per fissare le idee, supponiamo che $|\phi\rangle$ sia scritto nella base standard come segue:

$$|\phi\rangle = a|0\rangle + b|1\rangle$$

Come nel caso del *dense coding*, Fabbri e Magri condividono un sistema fisico composto da due qubit entangled e descritto dallo stato $|\psi\rangle$. Lo stato entangled condiviso è uno stato di Bell:

$$|\psi\rangle_{23} = \frac{1}{\sqrt{2}} \left(|0\rangle_2 \otimes |0\rangle_3 + |1\rangle_2 \otimes |1\rangle_3\right)$$

Il qubit 1 è quello che Fabbri vuole teletrasportare, i qubit 2 e 3 formano la coppia entangled condivisa, dove il qubit 2 è in possesso di Fabbri e il qubit 3 è in possesso di Magri. Lo stato fisico totale del sistema è quindi:

$$|\Phi_{\text{iniziale}}\rangle_{123} = |\phi\rangle_1 \otimes |\psi\rangle_{23} = (a|0\rangle_1 + b|1\rangle_1) \otimes \frac{1}{\sqrt{2}} \left(|0\rangle_2 \otimes |0\rangle_3 + |1\rangle_2 \otimes |1\rangle_3\right)$$

### 7.4.2 Preparazione e trasmissione dei dati

Il signor Fabbri esegue una misura di Bell sui qubit 1 e 2 in suo possesso. Per fare ciò, trasforma lo stato combinato dei qubit 1 e 2 agendo prima con un gate CNOT (con qubit 1 come controllo e qubit 2 come target) e poi con un operatore di Hadamard sul qubit 1. Più esplicitamente, egli applica:

$$|\Phi'\rangle_{123} = (H_1 \otimes I_2 \otimes I_3) \left(\text{CNOT}_{1\to 2} \otimes I_3\right) |\Phi_{\text{iniziale}}\rangle_{123}$$

Dopo queste operazioni, lo stato del sistema diventa:

$$|\Phi'\rangle_{123} = \frac{1}{2} \left(|00\rangle_{12}(a|0\rangle_3 + b|1\rangle_3) + |01\rangle_{12}(a|1\rangle_3 + b|0\rangle_3) + |10\rangle_{12}(a|0\rangle_3 - b|1\rangle_3) + |11\rangle_{12}(a|1\rangle_3 - b|0\rangle_3)\right)$$

Il signor Fabbri esegue quindi la misura dei suoi due qubit 1 e 2, ottenendo una delle quattro possibili combinazioni di risultati:

- $|00\rangle_{12}$
- $|01\rangle_{12}$
- $|10\rangle_{12}$
- $|11\rangle_{12}$

Egli trasmette i due bit classici corrispondenti ai risultati di misura al signor Magri.

### 7.4.3 Ricezione e decodifica

A questo punto, il qubit di Magri (qubit 3) si trova in uno stato che dipende dai risultati di misura di Fabbri:

- Se Fabbri ha ottenuto $|00\rangle_{12}$, allora il qubit di Magri è nello stato $a|0\rangle_3 + b|1\rangle_3$
- Se Fabbri ha ottenuto $|01\rangle_{12}$, allora il qubit di Magri è nello stato $a|1\rangle_3 + b|0\rangle_3$
- Se Fabbri ha ottenuto $|10\rangle_{12}$, allora il qubit di Magri è nello stato $a|0\rangle_3 - b|1\rangle_3$
- Se Fabbri ha ottenuto $|11\rangle_{12}$, allora il qubit di Magri è nello stato $a|1\rangle_3 - b|0\rangle_3$

Il signor Magri, avendo ricevuto i due bit classici da Fabbri, sa quale stato ha il suo qubit e può applicare una delle quattro trasformazioni per recuperare lo stato originale $|\phi\rangle$:

- Se riceve 00, non fa nulla (applica l'identità $I$)
- Se riceve 01, applica il gate $X$
- Se riceve 10, applica il gate $Z$
- Se riceve 11, applica il gate $X$ seguito da $Z$ (cioè il gate $Y = iXZ$)

Dopo la trasformazione appropriata, il qubit di Magri si trova esattamente nello stato originale:

$$|\phi\rangle = a|0\rangle_3 + b|1\rangle_3$$

Il qubit è stato quindi teletrasportato dal signor Fabbri al signor Magri attraverso la trasmissione di due bit classici.

La cosa veramente sorprendente è che, usando due bit classici, che possono codificare in totale solo quattro diversi possibili valori, si possono trasmettere i due coefficienti complessi $a$ e $b$, che possono assumere un'infinità di valori. Questo non viola il principio secondo cui da un qubit si può estrarre al massimo un bit di informazione, poiché l'informazione sui coefficienti $a$ e $b$ non viene trasmessa attraverso i canali classici, ma viene trasferita grazie all'entanglement e alle operazioni quantistiche.

Anche il *teletrasporto quantistico* è un elemento chiave nella sicurezza delle trasmissioni che può essere realizzata usando le tecnologie quantistiche.

### 7.4.4 Prime conclusioni

Con questo capitolo si è conclusa la presentazione dei concetti base dell'informatica quantistica. Come si è visto, i gate quantistici possono riprodurre completamente qualsiasi circuito logico, quindi possono in linea di principio rimpiazzare l'attuale tecnologia elettronica usata per il calcolo automatico (computing).

Usando i principi dell'*entanglement* e della *sovrapposizione degli stati*, è possibile andare oltre la computazione classica, realizzando circuiti che sfruttano appieno le peculiarità della meccanica quantistica che abbiamo qui esposto. Il risultato è di ottenere algoritmi più veloci per risolvere problemi complessi, come ad esempio l'algoritmo di Shor per la fattorizzazione di numeri primi o l'algoritmo di Grover per la ricerca non strutturata.

L'informatica quantistica è destinata a entrare prepotentemente nella società dell'informazione attuale. Aver compiuto questo primo percorso cognitivo nei suoi intimi segreti è un passo importante per restare allineati al progresso scientifico e tecnologico e per proseguire il percorso imparando anche a sfruttare le sue nuove caratteristiche.

Molte delle novità interessanti su questo argomento sono pubblicate in articoli scientifici come quelli riportati in bibliografia. Le nozioni qui presentate permetteranno di affrontarne la lettura.

## 7.5 Bibliografia Ragionata

Per approfondire i concetti trattati in questo capitolo e comprendere appieno le fondamenta teoriche e le applicazioni pratiche della computazione quantistica, è fondamentale consultare una serie di opere chiave nel campo.

Il testo di Nielsen and Chuang [2010], *Quantum Computation and Quantum Information*, è considerato uno dei riferimenti più autorevoli e completi sull'argomento. Gli autori offrono una trattazione dettagliata dei principi fondamentali della computazione quantistica, inclusi il teorema di non clonazione, il dense coding e il teletrasporto quantistico. Il libro fornisce anche approfondimenti sulle tecniche di correzione degli errori quantistici e sulle sfide associate alla realizzazione pratica dei computer quantistici.

Mermin [2007], nel suo *Quantum Computer Science: An Introduction*, presenta un'introduzione accessibile ma profonda alla computazione quantistica. Mermin spiega in modo chiaro e intuitivo concetti complessi come l'entanglement e i fenomeni non locali, fornendo esempi pratici che aiutano il lettore a comprendere le implicazioni del teorema di non clonazione e le potenzialità del dense coding e del teletrasporto.

Per una comprensione più approfondita delle basi matematiche e concettuali della meccanica quantistica, il libro di Peres [1995], *Quantum Theory: Concepts and Methods*, è una risorsa inestimabile. Peres esplora in dettaglio i principi fondamentali che sono alla base dei fenomeni quantistici sfruttati nella computazione quantistica, offrendo una prospettiva rigorosa e completa.

Sakurai and Napolitano [2014], con *Modern Quantum Mechanics*, offre una trattazione avanzata dei concetti fondamentali della meccanica quantistica, con enfasi sulle applicazioni moderne. Il libro approfondisce temi come l'entanglement, le proprietà degli operatori e le trasformazioni unitarie, fornendo gli strumenti matematici necessari per comprendere appieno protocolli come il teletrasporto quantistico.

Le *Lectures on Physics* di Feynman et al. [2011], in particolare il volume 3 dedicato alla meccanica quantistica, rappresentano un classico intramontabile. Feynman, con il suo

stile inconfondibile, offre intuizioni profonde sui principi quantistici fondamentali, enfatizzando l'importanza dei fenomeni di sovrapposizione e entanglement. Queste lezioni sono particolarmente utili per comprendere le basi concettuali del teorema di non clonazione e delle altre peculiarità della computazione quantistica.

Infine, l'articolo di Bennett et al. [1993], *Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels*, è fondamentale per comprendere il protocollo del teletrasporto quantistico. In questo lavoro pionieristico, gli autori descrivono in dettaglio come è possibile trasferire uno stato quantistico sconosciuto utilizzando l'entanglement e la comunicazione classica, aprendo nuove prospettive nella trasmissione dell'informazione quantistica.

Queste opere, combinando teoria e pratica, forniscono al lettore una comprensione completa dei principi fondamentali della computazione quantistica e delle sue applicazioni rivoluzionarie. Approfondire questi testi permetterà di apprezzare non solo le potenzialità ma anche le sfide associate allo sviluppo di tecnologie quantistiche avanzate.

## 7.6 Esercizi pratici con QuantumSim

Per consolidare i concetti di dense coding e teletrasporto quantistico, utilizziamo **QuantumSim** per simulare i protocolli descritti nel capitolo.

### 7.6.1 Simulazione del dense coding

Il dense coding permette di trasmettere due bit classici di informazione usando un solo qubit. Implementiamo le trasformazioni di Fabbri e la decodifica di Magri.

```
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
    QubitState* state = initializeState(2); // Inizializza due qubit
    entangled

    // Stato entangled iniziale
    applyHadamard(state, 0); // Crea la sovrapposizione
    applyCNOT(state, 0, 1);  // Entanglement con il CNOT
    printf("Stato iniziale entangled:\n");
    printState(state);

    // Fabbri applica una trasformazione sul suo qubit
    applyX(state, 0); // Ad esempio, trasforma secondo il bit classico "Sud
    "
    printf("Stato dopo la trasformazione di Fabbri:\n");
    printState(state);

    // Magri decodifica
    applyCNOT(state, 0, 1); // CNOT per la decodifica
```

```
    applyHadamard(state, 0); // Hadamard per la decodifica

    // Magri misura i due qubit
    MeasurementResult result1 = measure(state, 0);
    MeasurementResult result2 = measure(state, 1);
    printf("Risultati delle misure: %d, %d\n", result1.result, result2.
    result);

    // Libera la memoria allocata
    freeState(state);
    return 0;
}
```

Listing 7.1: circuit7.1.c

**Spiegazione:**

- Lo stato iniziale è entangled.
- Fabbri applica una trasformazione ($I$, $X$, $Z$, o $iY$) per codificare il suo messaggio.
- Magri usa il CNOT e l'Hadamard per decodificare e misura i qubit per ottenere i due bit classici trasmessi.

### 7.6.2 Simulazione del teletrasporto quantistico

Il teletrasporto quantistico permette di trasferire lo stato di un qubit usando un canale classico e uno stato entangled condiviso.

```
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
    QubitState* state = initializeState(3); // Tre qubit: 1 da
    teletrasportare, 2 entangled

    // Stato del qubit da teletrasportare
    initializeStateTo(state, 0); // Inizializza il primo qubit
    applyHadamard(state, 0);     // Stato iniziale da teletrasportare

    // Stato entangled tra il secondo e il terzo qubit
    applyHadamard(state, 1);
    applyCNOT(state, 1, 2);
    printf("Stato iniziale:\n");
    printState(state);

    // Misura di Bell sui primi due qubit
    applyCNOT(state, 0, 1);
    applyHadamard(state, 0);
    MeasurementResult result1 = measure(state, 0);
    MeasurementResult result2 = measure(state, 1);
```

```
22      printf("Risultati della misura di Bell: %d, %d\n", result1.result,
        result2.result);
23
24      // Magri applica trasformazioni basate sui risultati
25      if (result1.result == 1) applyX(state, 2);
26      if (result2.result == 1) applyZ(state, 2);
27
28      printf("Stato del qubit teletrasportato:\n");
29      printState(state);
30
31      // Libera la memoria allocata
32      freeState(state);
33      return 0;
34  }
```

Listing 7.2: circuit7.2.c

**Spiegazione:**

- Il qubit da teletrasportare viene inizializzato in uno stato noto.
- Fabbri esegue la misura di Bell e trasmette i risultati a Magri.
- Magri applica trasformazioni ($X$, $Z$) per recuperare lo stato originale sul suo qubit.

## Conclusione

Con questi esercizi, abbiamo dimostrato come QuantumSim possa essere utilizzato per simulare protocolli avanzati di comunicazione quantistica. Nel **Capitolo 8**, approfondiremo gli algoritmi quantistici, che sfruttano i fenomeni di entanglement e sovrapposizione per ottenere vantaggi computazionali rispetto ai metodi classici.

# Capitolo 8

# Il computer quantistico

## 8.1 Introduzione

Il campo della computazione quantistica rappresenta una delle frontiere più affascinanti e promettenti della scienza e della tecnologia moderne. L'obiettivo fondamentale di un **computer quantistico** è quello di implementare fisicamente gli stati dei qubit e le trasformazioni unitarie che operano su di essi. Ciò implica la necessità di realizzare non solo un modello teorico, ma uno strumento concreto che permetta di manipolare e controllare qubit all'interno di un sistema fisico reale.

La costruzione di un computer quantistico comporta sfide significative sia dal punto di vista concettuale che tecnologico. Si tratta di progettare e assemblare un sistema fisico composto da registri di qubit e da una strumentazione in grado di eseguire operazioni quantistiche con elevata precisione e affidabilità. Questo richiede una comprensione profonda dei principi della meccanica quantistica e l'abilità di tradurre tali principi in tecnologie pratiche.

Per affrontare queste sfide in modo sistematico, il fisico David DiVincenzo ha proposto una lista di criteri fondamentali che un sistema fisico deve soddisfare per essere considerato un computer quantistico funzionante. Questi criteri, noti come **criteri di DiVincenzo**, forniscono una guida per lo sviluppo di tecnologie quantistiche e aiutano a identificare le aree chiave su cui concentrare la ricerca e l'innovazione.

In questo capitolo, esploreremo in dettaglio:

- **I criteri di DiVincenzo**: Analizzeremo ciascuno dei sette criteri, discutendo la loro importanza e le implicazioni per la realizzazione pratica di un computer quantistico.
- **Computer quantistico fotonico**: Approfondiremo il caso specifico dei computer quantistici basati su fotoni, esaminando i vantaggi e le sfide associati a questa tecnologia.
- **Architettura di un computer quantistico**: Descriveremo i componenti fondamentali di un computer quantistico e come essi interagiscono per eseguire calcoli quantistici.

- **Programmazione quantistica**: Discuteremo come si programma un computer quantistico, evidenziando il ruolo cruciale dell'interazione tra software classico e hardware quantistico.

Questo capitolo mira a fornire una comprensione approfondita di ciò che è necessario per costruire un computer quantistico funzionante e delle sfide attuali nel campo. Esploreremo sia gli aspetti teorici che quelli pratici, offrendo una panoramica delle tecnologie emergenti e delle direzioni future della ricerca.

## 8.2 I criteri di DiVincenzo

I criteri di DiVincenzo sono suddivisi in due gruppi. I primi cinque concernono la computazione quantistica:

1. Un sistema fisico scalabile con qubit ben caratterizzati.
2. La capacità di inizializzare lo stato dei qubit in uno stato semplice e ben definito.
3. Tempi di decoerenza significativamente più lunghi dei tempi di operazione delle porte logiche.
4. Un insieme universale di porte logiche quantistiche.
5. Una capacità di misura specifica per i qubit.

I successivi due criteri riguardano la comunicazione quantistica:

6. La capacità di interconnettere qubit stazionari e volanti (cioè la possibilità di trasferire l'informazione quantistica tra qubit diversi).
7. La capacità di trasmettere qubit volanti tra luoghi specificati con alta fedeltà.

Dall'inizio del XXI secolo ad oggi (2021), questa sfida è stata raccolta da diverse organizzazioni sia a scopo di ricerca che commerciale. Sono stati costruiti diversi prototipi funzionanti che ne hanno dimostrato la fattibilità. Ciò nonostante, la realizzazione di un computer quantistico rimane ancora una vera sfida perché ognuno dei punti della lista di DiVincenzo presenta aspetti complessi. Vediamoli nel dettaglio.

## 8.3 Computer quantistico fotonico

In questo testo lo stato di un qubit è sempre stato associato allo stato di polarizzazione di un fotone. Ci sono due motivi per questa scelta. Da un lato, per quanto il fotone sia una particella priva di massa e quindi sfuggente all'intuizione comune, il concetto di polarizzazione e di lente polarizzata è molto comune, in quanto se ne fa esperienza già con gli occhiali da sole o con gli occhiali per la visione in tre dimensioni. Dall'altro, allo

stato attuale della tecnologia, i computer quantistici basati sulla fotonica sembrano quelli che possono mantenere più facilmente la promessa di essere integrati su chip e scalare in dimensione. In ogni modo, per coerenza con il resto del testo, in questo capitolo viene fornita una panoramica delle problematiche e delle possibilità della realizzazione di un computer quantistico di tipo fotonico.

Il qubit è un sistema fisico, naturale o artificiale, caratterizzato da due stati fisici ben distinti. Lo stato di polarizzazione di un fotone, ampiamente utilizzato in questo testo, è un esempio di sistema fisico a due stati; per questo motivo è possibile pensare a un computer quantistico basato su qubit di tipo fotonico.

Allo stato attuale esistono già processori quantistici come quelli sviluppati da Xanadu™ che utilizzano questa tecnologia realizzata direttamente su chip di silicio. Per fare un primo passo nell'architettura di un processore quantistico, vediamo come i criteri visti sopra sono realizzati nella sua declinazione fotonica.

1. **Qubit ben caratterizzati e scalabili**: Usando i fotoni come qubit, un processore quantistico fotonico dispone già di un sistema fisico a due stati ben caratterizzati. Tuttavia, i sistemi attuali come quelli di Xanadu non usano qubit basati su singoli fotoni, ma sfruttano stati quantistici continui detti *stati squeezed* (compressi), tipici dell'ottica quantistica continua.

2. **Inizializzazione dello stato dei qubit**: Inizializzare un qubit codificato nella polarizzazione di un fotone è una sfida più ardua rispetto all'inizializzazione di un bit classico. Il problema risiede nella natura intrinsecamente quantistica del fotone. L'unico modo per essere sicuri di aver prodotto un fotone è rilevarlo con un apparecchio, ma in questo caso il fotone viene assorbito e quindi non è più utile come qubit. La rivelazione di un fotone con un filtro polarizzatore ci permette di conoscerne lo stato di polarizzazione, ma al contempo si perde il fotone stesso. D'altra parte, la mancata rivelazione di un fotone da parte di un polarizzatore ci lascia in una situazione di incertezza: il fotone potrebbe essere passato attraverso perché si trova in un dato stato di polarizzazione, oppure potrebbe non essere stato emesso affatto.

   Come si può capire, il problema dell'inizializzazione dei qubit fotonici è alquanto sfidante. Una possibile soluzione per preparare (inizializzare) i qubit in uno stato definito (per esempio $|0\rangle$ o $|1\rangle$) è utilizzare la tecnica detta *spontaneous parametric down-conversion* (SPDC), che consiste nel trasformare un fotone ad alta energia in una coppia di fotoni a più bassa energia. La caratteristica di questo processo è che i due fotoni emessi sono entangled, quindi dalla polarizzazione di uno è possibile risalire alla polarizzazione dell'altro (vedi paragrafo 4.15). In questo modo, uno dei due fotoni può essere misurato e sacrificato per conoscere la polarizzazione dell'altro. A questo punto, in base alla polarizzazione voluta, si mantiene il fotone come è oppure si trasforma otticamente in modo da cambiarne lo stato di polarizzazione.

3. **Tempi di decoerenza**: Il concetto di decoerenza, o perdita di coerenza, è molto complesso e difficilmente è possibile farsene un'idea intuitiva senza entrare nei dettagli dei calcoli. Tuttavia, è bene indagarlo almeno a livello descrittivo.

   Nel capitolo 4 è stata introdotta la funzione d'onda e si è visto che essa deve descrivere completamente il sistema considerato. Si consideri ora un sistema complesso, suddivisibile in più sottosistemi; per esempio, possiamo pensare a una molecola composta da diversi atomi. Ogni atomo è un sottosistema del sistema molecola. Dalla meccanica quantistica possiamo aspettarci che esista una funzione d'onda $\psi_{\text{molecola}}(x)$ che descriva completamente la molecola. Supponiamo ora di voler studiare un atomo della molecola. Potremmo essere indotti a pensare di studiare la funzione $\psi_{\text{atomo}}(x)$, ma questo in generale non sarebbe corretto perché con ogni probabilità l'atomo oggetto di studio (o i suoi costituenti) si trova in entanglement con gli altri atomi; quindi, non è possibile isolarne una specifica funzione d'onda per descriverlo.

   Se si considera il momento in cui gli atomi si legano per formare la molecola, si trova un istante in cui essi non erano ancora entangled e quindi potevano essere descritti singolarmente ognuno dalla propria funzione d'onda.

   Un ragionamento analogo può valere per un sistema costituito da un fotone e una serie di atomi. In pratica, per un sistema complesso, i singoli costituenti non sono descrivibili in termini di funzione d'onda individuale, esattamente come per due fotoni entangled non esistono separatamente la funzione d'onda dell'uno e dell'altro. Ciò detto, un computer quantistico deve operare su qubit che devono essere descritti da funzioni d'onda isolate; quindi, quando viene preparato un qubit, deve essere *libero* da entanglement e in uno stato ben definito. Il problema è che, in breve tempo, il qubit entrerà in *contatto* con il resto dell'ambiente (ad esempio l'elettronica del computer quantistico stesso), inizierà a *interagire* e a perdere la purezza del suo stato per entrare in entanglement con il resto del sistema.

   Prima che ciò avvenga, il computer quantistico deve aver sfruttato il qubit. Il tempo di coerenza richiesto è quindi un compromesso tra il tempo *operativo* dei gate che devono agire sul qubit e il tempo per il quale si può supporre che il qubit non abbia interagito con l'ambiente.

   Alcuni computer a *ioni intrappolati* sfruttano come qubit i due livelli energetici definiti dalla separazione *iperfine* dovuta all'interazione degli spin degli elettroni con quello nucleare. Il tempo di decoerenza di questi stati può essere dell'ordine di secondi o anche più lungo, il che è vantaggioso per le operazioni quantistiche. Altre tecnologie sfruttano stati di Rydberg o livelli energetici elettronici, con tempi di decoerenza più brevi ma comunque sufficienti rispetto ai tempi *operativi* dei gate quantistici.

   Per quanto riguarda la realizzazione di computer fotonici, bisogna tenere conto che i fotoni viaggiano alla velocità della luce, quindi il tempo operativo dei gate deve essere molto ridotto; il problema si riduce quindi a incanalare il fotone nel gate prima che

abbia perso la sua coerenza. Questo però non è un problema di poco conto. I computer fotonici su chip attualmente in produzione non usano singoli fotoni per codificare i qubit, ma stati di luce *squeezed*, che permettono di gestire meglio la decoerenza e sfruttano l'ottica quantistica a variabili continue.

4. **Un sistema di gate universale**: Un sistema di gate universale che agisca sui fotoni è, in linea di principio, realizzabile utilizzando componenti di ottica lineare come specchi, beam splitter e phase shifter. Le trasformazioni su qubit singoli sono relativamente semplici da implementare. Tuttavia, la realizzazione di interazioni a due qubit (come il gate CNOT), necessarie per avere un set di gate universale, è più complessa con i fotoni, poiché richiede interazioni non lineari o l'uso di misure proiettive e fotoni ancillari.

   Sebbene l'idea di realizzare computer solo con ottica lineare sia interessante, questa linea di ricerca è limitata dalla difficoltà di scalare il sistema a un gran numero di qubit. Le soluzioni più promettenti tendono a sfruttare le proprietà dell'ottica non lineare per produrre circuiti quantistici integrati su silicio.

5. **Capacità di misurare i qubit**: La misura dei qubit fotonici può essere effettuata con rivelatori di fotoni singoli ad alta efficienza. Tuttavia, la rilevazione efficiente dei fotoni è una sfida tecnologica, poiché richiede dispositivi con basse perdite e rumore minimo.

6. **Interconnessione tra qubit di calcolo e qubit volanti**: La possibilità di trasformare un qubit di calcolo in un qubit di comunicazione è il corrispondente quantistico del lavoro che fanno le nostre schede di rete sui bit classici quando ci colleghiamo a Internet con il PC o con il telefono. Lo stesso deve saper fare un computer quantistico se lo si vuole collegare a una rete di computer quantistici.

   In linea di principio, la trasformazione tra qubit di calcolo e qubit di comunicazione avviene naturalmente nella computazione fotonica. Infatti, un fotone che è stato manipolato in un circuito quantistico può essere inviato così com'è lungo una linea ottica (per esempio una fibra ottica) e raggiungere un altro computer quantistico.

   In teoria sembra semplice, ma nella realtà le cose sono più complesse. Infatti, come accennato, le soluzioni fotoniche attualmente usate sono basate su stati squeezed, per i quali il tempo di coerenza non è idealmente infinito come per un singolo fotone, e possono essere soggetti a perdite e rumore durante la trasmissione.

7. **Trasmissione affidabile dei qubit**: I qubit fotonici possono essere trasmessi facilmente sia nel vuoto che lungo una linea ottica. Il problema è ovviamente quello di mantenere lo stato di coerenza del qubit finché non raggiunge la destinazione. Le perdite nelle fibre ottiche, il rumore e la dispersione possono influenzare negativamente la trasmissione dei qubit fotonici. Tecniche come la correzione degli errori

quantistici e l'uso di ripetitori quantistici sono oggetto di ricerca per superare queste sfide.

## 8.4 Architettura di un computer quantistico

La realizzazione di un computer quantistico richiede l'integrazione di sistemi quantistici delicati con apparati di controllo classici complessi. Un'architettura tipica di un computer quantistico comprende diversi componenti fondamentali:

- **Array di qubit**: un insieme di qubit fisici che costituiscono il cuore del computer quantistico.
- **Elettronica quantistica di controllo**: hardware specializzato che genera i segnali necessari per manipolare e misurare i qubit.
- **Elettronica di controllo classica**: sistemi di calcolo classici che orchestrano le operazioni quantistiche, elaborano i dati di misura e gestiscono l'interfaccia utente.
- **Sistema di correzione degli errori quantistici**: implementa codici di correzione per proteggere l'informazione quantistica dagli errori.
- **Memoria quantistica**: permette di immagazzinare stati quantistici per periodi di tempo prolungati.
- **Interfaccia di comunicazione quantistica**: consente la trasmissione di qubit tra diversi sistemi o all'interno di una rete quantistica.
- **Infrastruttura criogenica (se necessaria)**: per tecnologie che richiedono temperature estremamente basse, come i qubit superconduttori.

Di seguito, descriviamo in dettaglio ciascuno di questi componenti e il loro ruolo nell'architettura complessiva.

### 8.4.1 Array di qubit

L'**array di qubit** è l'insieme dei qubit fisici utilizzati per eseguire i calcoli quantistici. La scelta della tecnologia per i qubit dipende da vari fattori, tra cui la coerenza quantistica, la scalabilità e la facilità di controllo. Alcune delle tecnologie più comuni includono:

- **Qubit superconduttori**: basati su giunzioni Josephson, operano a temperature prossime allo zero assoluto (circa 10 millikelvin). Sono controllati da impulsi di microonde e sono attualmente utilizzati da aziende come IBM, Google e Rigetti.
- **Ioni intrappolati**: atomi ionizzati intrappolati in campi elettrici o magnetici, manipolati con laser. Offrono tempi di coerenza lunghi e alta fedeltà nelle operazioni.

- **Qubit fotonici**: utilizzano fotoni come portatori di informazione quantistica, manipolati attraverso circuiti ottici integrati. Possono operare a temperatura ambiente e sono promettenti per la comunicazione quantistica.
- **Qubit a spin di elettroni o nuclei**: utilizzano lo spin di elettroni o nuclei in materiali come il silicio o il diamante (centri di vacanza di azoto). Possono essere controllati con campi magnetici e hanno il potenziale per essere integrati con la tecnologia dei semiconduttori.
- **Qubit topologici**: basati su quasiparticelle e promettono una maggiore resistenza alla decoerenza.

### 8.4.2 Elettronica quantistica di controllo

L'**elettronica quantistica di controllo** è responsabile della generazione dei segnali necessari per manipolare e leggere lo stato dei qubit. Questo include:

- **Generazione di impulsi di controllo**: per manipolare i qubit, vengono utilizzati impulsi di microonde, laser o campi magnetici a radiofrequenza, a seconda della tecnologia dei qubit.
- **Sistemi di lettura (readout)**: per misurare lo stato dei qubit, sono necessari rivelatori sensibili che possano distinguere tra gli stati quantistici con alta fedeltà.
- **Filtraggio e schermatura**: per proteggere i qubit dal rumore esterno e dalle interferenze elettromagnetiche, si utilizzano filtri a basso rumore e schermature metalliche.
- **Criogenia**: nel caso di qubit superconduttori o spin elettronici a basse temperature, l'hardware di controllo deve operare a temperature estremamente basse, richiedendo sistemi di refrigerazione avanzati.

### 8.4.3 Elettronica di controllo classica

L'**elettronica di controllo classica** comprende i sistemi di calcolo convenzionali che orchestrano l'esecuzione degli algoritmi quantistici. Questi sistemi svolgono le seguenti funzioni:

- **Sequenziamento delle operazioni**: generano le sequenze di impulsi di controllo necessari per implementare le porte quantistiche e gli algoritmi desiderati. Questo richiede hardware programmabile ad alta velocità, come FPGA (Field-Programmable Gate Array) o DSP (Digital Signal Processor).
- **Elaborazione dei dati di misura**: raccolgono e analizzano i risultati delle misure dei qubit, interpretando gli esiti probabilistici tipici dei sistemi quantistici.

- **Correzione degli errori quantistici**: implementano algoritmi di correzione degli errori per mitigare gli effetti della decoerenza e delle imperfezioni hardware. Questo include la gestione di qubit ancillari e la decodifica degli errori.
- **Interfaccia utente e software**: forniscono gli strumenti software per programmare il computer quantistico, compilare gli algoritmi quantistici e visualizzare i risultati. Questo comprende anche livelli di astrazione più alti, come linguaggi di programmazione quantistica (ad esempio, Qiskit, Cirq, Forest o appunto QuantumSim).

### 8.4.4 Sistema di correzione degli errori quantistici

La **correzione degli errori quantistici** è essenziale per il funzionamento affidabile dei computer quantistici. Poiché i qubit sono estremamente sensibili al rumore e alla decoerenza, è necessario implementare codici di correzione degli errori per proteggere l'informazione quantistica. Componenti chiave includono:

- **Qubit fisici e logici**: i qubit fisici sono combinati per formare qubit logici che possono resistere a errori grazie alla ridondanza e all'uso di codici quantistici come il codice di superficie o il codice di stabilizzatore.
- **Misure ancillari**: qubit ancillari sono utilizzati per rilevare errori senza distruggere lo stato quantistico dei qubit logici.
- **Decodificatori**: algoritmi che interpretano i risultati delle misure ancillari per determinare la presenza e il tipo di errori, permettendo le correzioni appropriate.

### 8.4.5 Memoria quantistica

La **memoria quantistica** consente di immagazzinare stati quantistici per periodi di tempo prolungati, permettendo la sincronizzazione tra diverse parti del calcolo o la comunicazione quantistica differita. Caratteristiche importanti includono:

- **Tempi di coerenza lunghi**: la memoria deve mantenere la coerenza degli stati quantistici per tempi sufficienti rispetto alle operazioni del calcolo.
- **Interfaccia con qubit attivi**: deve essere possibile trasferire stati quantistici tra la memoria e i qubit attivi nel processore.
- **Scalabilità**: la memoria dovrebbe essere scalabile per supportare applicazioni che richiedono la memorizzazione di molti qubit.

### 8.4.6 Interfaccia di comunicazione quantistica

L'**interfaccia di comunicazione quantistica** è responsabile della trasmissione di qubit tra diversi sistemi o nodi all'interno di una rete quantistica. Questo componente è fondamentale per:

- **Reti quantistiche**: collegare diversi computer quantistici per formare un cluster o una rete distribuita.
- **Teletrasporto quantistico**: trasferire stati quantistici senza spostare fisicamente il qubit, utilizzando l'entanglement e canali classici.
- **Ripetitori quantistici**: estendere la distanza di trasmissione dei qubit attraverso canali rumorosi o attenuanti.

Tecnologie coinvolte includono l'uso di fotoni come qubit volanti, interfacce tra qubit stazionari (come qubit superconduttori o ioni intrappolati) e qubit volanti, e l'implementazione di protocolli di comunicazione quantistica sicura.

### 8.4.7 Infrastruttura criogenica

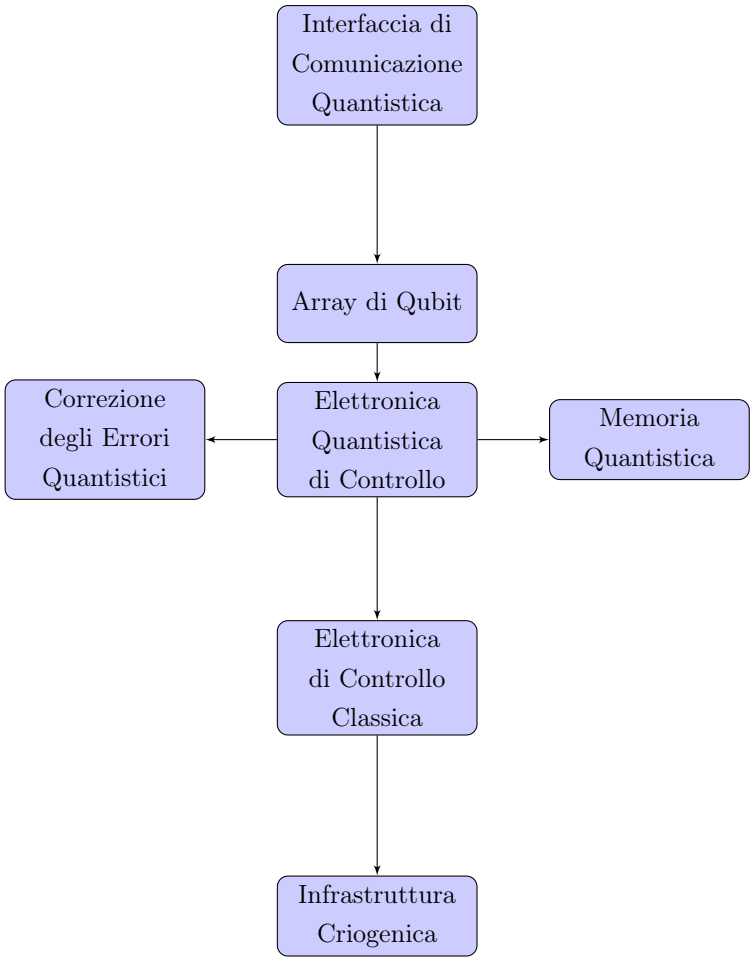
Per alcune tecnologie di qubit, come i **qubit superconduttori**, è necessaria un'**infrastruttura criogenica** per mantenere le temperature estremamente basse richieste per il loro funzionamento (tipicamente intorno ai 10 millikelvin). Questa include:

- **Frigoriferi a diluizione**: apparecchiature che utilizzano una miscela di isotopi di elio per raggiungere temperature prossime allo zero assoluto.
- **Isolamento termico**: sistemi per minimizzare l'assorbimento di calore dall'ambiente esterno.
- **Gestione dei cablaggi criogenici**: utilizzo di materiali e design speciali per i cavi che collegano l'elettronica di controllo classica ai qubit, minimizzando la conduzione del calore e il rumore elettrico.

### 8.4.8 Interazione tra componenti

La figura seguente illustra l'architettura di un computer quantistico e le interazioni tra i suoi componenti principali.

Interazione con Reti Quantistiche




Isolamento, Gestione Termica, Sincronizzazione e Scalabilità

In questa architettura:

- L'**array di qubit** costituisce il processore quantistico dove avvengono le operazioni quantistiche.
- L'**elettronica quantistica di controllo** interagisce direttamente con i qubit per eseguire le operazioni e le misure.
- Il **sistema di correzione degli errori quantistici** utilizza qubit ancillari e misure per proteggere l'informazione.
- La **memoria quantistica** è collegata all'array di qubit per l'immagazzinamento degli stati.

- L'**interfaccia di comunicazione quantistica** permette lo scambio di qubit con altri sistemi o reti.
- L'**elettronica di controllo classica** coordina tutte le operazioni, gestisce gli algoritmi di alto livello e processa i risultati.
- L'**infrastruttura criogenica** assicura le condizioni operative necessarie per i qubit sensibili alla temperatura.

La progettazione di un computer quantistico richiede un'attenta considerazione di come queste componenti interagiscono tra loro, specialmente in termini di:

- **Isolamento dai disturbi esterni**: schermature fisiche ed elettroniche per proteggere i qubit dal rumore ambientale.
- **Gestione termica**: controllo delle temperature per mantenere le condizioni operative ottimali e minimizzare le fluttuazioni termiche.
- **Sincronizzazione**: coordinamento temporale preciso tra le diverse operazioni quantistiche e classiche.
- **Scalabilità**: progettazione modulare che permetta di aumentare il numero di qubit senza degradare le prestazioni.

### 8.4.9 Sfide tecnologiche

La realizzazione pratica di un computer quantistico presenta numerose sfide:

- **Scalabilità**: aumentare il numero di qubit mantenendo la qualità delle operazioni e la gestione degli errori. Questo richiede innovazioni nell'integrazione dei circuiti e nella riduzione delle interferenze tra qubit.
- **Coerenza**: prolungare i tempi di coerenza dei qubit per eseguire calcoli più complessi. Si stanno esplorando nuovi materiali e tecniche di isolamento per migliorare la coerenza.
- **Correzione degli errori**: sviluppare codici di correzione degli errori efficienti che siano tolleranti alle imperfezioni hardware. Questo include la ricerca di codici con soglia di errore più alta e la riduzione dell'overhead in termini di qubit aggiuntivi.
- **Integrazione hardware-software**: creare strumenti di programmazione e compilatori che possano tradurre algoritmi quantistici in sequenze di operazioni hardware ottimizzate, tenendo conto delle limitazioni fisiche del sistema.
- **Interconnessione**: sviluppare metodi per collegare tra loro diversi computer quantistici o per implementare architetture modulari. Questo coinvolge la ricerca in fotonica integrata e interfacce quantistiche ibride.

- **Affidabilità e riproducibilità**: garantire che i sistemi quantistici funzionino in modo prevedibile e possano essere prodotti su larga scala con caratteristiche uniformi.
- **Costi e infrastruttura**: ridurre i costi associati alla costruzione e al mantenimento dei computer quantistici, specialmente per quanto riguarda le infrastrutture criogeniche e l'elettronica di controllo avanzata.

Nonostante queste sfide, i progressi nella ricerca e nello sviluppo tecnologico stanno portando rapidamente verso la realizzazione di computer quantistici sempre più potenti e affidabili. Collaborazioni interdisciplinari tra fisici, ingegneri, informatici e matematici sono essenziali per superare gli ostacoli e sfruttare appieno il potenziale della computazione quantistica.

### 8.4.10 La sfida di un computer completamente quantistico senza elettronica classica

Nonostante i progressi significativi nella realizzazione di componenti quantistici, attualmente è estremamente difficile, se non impossibile, costruire un computer completamente quantistico che non faccia uso di elettronica classica. La principale ragione di questa difficoltà risiede nelle sfide tecnologiche e teoriche che emergono quando si cerca di sostituire le funzioni svolte dall'elettronica classica con equivalenti quantistici.

#### Controllo e orchestrazione delle operazioni quantistiche

Le operazioni quantistiche richiedono una precisione e un controllo estremamente elevati. Attualmente, l'elettronica classica è indispensabile per generare i segnali di controllo (come impulsi di microonde, laser o campi magnetici) necessari per manipolare i qubit. Questi segnali devono essere sincronizzati con precisione e modulati in modo complesso, compiti per i quali l'elettronica classica è altamente sviluppata e affidabile.

Sostituire questi sistemi con controlli completamente quantistici richiederebbe dispositivi quantistici che possano generare e manipolare segnali con la stessa precisione, il che al momento è al di là delle nostre capacità tecnologiche.

#### Misura e lettura degli stati quantistici

La misurazione degli stati dei qubit è un processo che, per sua natura, coinvolge un'interazione tra il sistema quantistico e un sistema classico. Gli esiti delle misure quantistiche devono essere registrati e interpretati, compiti che richiedono dispositivi classici. Inoltre, i risultati delle misure sono tipicamente probabilistici, e l'elaborazione statistica dei dati richiede risorse di calcolo classiche.

#### Correzione degli errori e feedback

La correzione degli errori quantistici spesso richiede un feedback rapido tra le misure dei qubit e le operazioni successive. Questo feedback loop è attualmente implementato uti-

lizzando elettronica classica, che può elaborare i risultati delle misure e determinare le correzioni necessarie in tempi molto brevi.

Implementare un sistema di correzione degli errori completamente quantistico implicherebbe la capacità di elaborare informazioni quantistiche (inclusi i risultati delle misure) senza mai convertirle in forma classica, il che è estremamente complesso e ancora oggetto di ricerca teorica.

#### Interfaccia con l'utente e programmazione

Gli utenti interagiscono con i computer quantistici attraverso interfacce e software che sono, per ora, interamente classici. La compilazione di algoritmi quantistici, l'invio di comandi al computer quantistico e la visualizzazione dei risultati sono tutte funzioni gestite da sistemi classici.

### 8.4.11 Prospettive future

Per superare queste limitazioni, la comunità scientifica sta esplorando diverse strade, tra cui:

- **Co-design quantistico-classico**: sviluppare architetture in cui le componenti quantistiche e classiche sono strettamente integrate, ottimizzando la comunicazione e minimizzando le latenze.
- **Dispositivi quantistici ibridi**: utilizzare sistemi che combinano proprietà quantistiche e classiche per svolgere funzioni di controllo e misura in modo più efficiente.
- **Tecnologie di controllo quantistico**: ricerca su metodi per implementare alcune funzioni di controllo utilizzando fenomeni quantistici, come l'uso di qubit ancillari per il controllo di altri qubit.

Tuttavia, è importante riconoscere che, almeno nel futuro prevedibile, l'elettronica classica continuerà a svolgere un ruolo cruciale nei computer quantistici. La sinergia tra sistemi quantistici e classici permette di sfruttare il meglio di entrambi i mondi: la potenza di calcolo dei qubit e la robustezza e versatilità dei sistemi classici.

### 8.4.12 Conclusioni

La costruzione di un computer completamente quantistico senza elettronica classica presenta sfide enormi sia dal punto di vista tecnologico che teorico. Mentre la ricerca continua a progredire, l'approccio più pragmatico è quello di sviluppare sistemi quantistici che lavorano in tandem con l'elettronica classica, sfruttando le capacità consolidate di quest'ultima per controllare, misurare e interagire con i sistemi quantistici. Questo approccio ibrido è attualmente il più promettente per avanzare nella realizzazione di computer quantistici pratici e utili.

## 8.5 Programmazione

Quanto visto finora ha chiarito come sia possibile costruire un circuito quantistico che esegue una computazione, però non si è ancora visto come fare a programmarlo, cioè stabilire la sequenza di trasformazioni quantistiche che devono aver luogo attraverso la scrittura di un programma.

Programmare un circuito quantistico significa configurare i quantum gate, per esempio stabilire l'angolo di rotazione di un operatore. Quindi, a livello concreto, significa definire una precisa sequenza di operazioni unitarie che possono avvenire su uno o più qubit.

Nel prossimo capitolo si vedrà un esempio concreto di programma quantistico, e quindi di come provare anche sperimentalmente a creare un circuito quantistico. La cosa che si vuole chiarire in questo paragrafo è cosa succede realmente quando si esegue un programma quantistico.

È importante notare che, ad oggi, non esiste un computer quantistico che sia completamente indipendente dalla computazione classica. Quando si esegue un programma quantistico, è presente un'elettronica (computer) classica che legge e interpreta il programma. Questo è scritto usando bit classici e l'elettronica, insieme al software classico, traduce il programma in azioni di configurazione del circuito quantistico.

In pratica, la programmazione in sé è classica, la configurazione del circuito è classica, mentre l'esecuzione è quantistica.

### 8.5.1 L'interazione tra programmazione classica e computazione quantistica

La programmazione di un computer quantistico avviene tipicamente attraverso linguaggi di programmazione classici che includono estensioni per definire circuiti quantistici. Esempi di questi linguaggi e framework includono Qiskit (Python), Cirq (Python), Q# (Microsoft), e altri.

Il programma quantistico è scritto su un computer classico e definisce la sequenza di operazioni quantistiche (gate) da applicare ai qubit. Queste operazioni sono poi tradotte in segnali di controllo che l'hardware classico invia all'hardware quantistico per manipolare i qubit.

### 8.5.2 Esecuzione di un programma quantistico

Quando si esegue un programma quantistico, il software classico invia le istruzioni all'hardware quantistico. L'hardware di controllo classico genera i segnali necessari (come impulsi di microonde, laser, ecc.) per eseguire le operazioni quantistiche specificate sul registro di qubit.

Dopo l'esecuzione del circuito quantistico, i risultati sono misurati, e i dati di misura (bit classici) sono inviati al computer classico per l'analisi. Poiché le misure quantistiche

sono probabilistiche, è spesso necessario eseguire il circuito molte volte per ottenere una distribuzione statistica dei risultati.

### 8.5.3 Il ruolo del software classico nella computazione quantistica

Il software classico ha un ruolo fondamentale nella computazione quantistica attuale. Esso si occupa di:

- **Definire i circuiti quantistici**: specificare quali gate quantistici applicare e in quale ordine.
- **Compilare il circuito**: tradurre le operazioni ad alto livello in istruzioni comprensibili dall'hardware quantistico specifico.
- **Ottimizzazione**: migliorare il circuito quantistico per ridurre il numero di gate, minimizzare l'errore e adattarsi alle limitazioni hardware.
- **Gestione delle risorse**: allocare i qubit fisici, tenendo conto della connettività e delle caratteristiche del dispositivo quantistico.
- **Raccolta e analisi dei risultati**: elaborare i dati ottenuti dalle misure per estrarre l'informazione desiderata.

### 8.5.4 Esempi di linguaggi e strumenti per la programmazione quantistica

Esistono diversi linguaggi di programmazione e strumenti software sviluppati per facilitare la programmazione di computer quantistici:

- **Qiskit**: un framework open-source sviluppato da IBM per programmare i computer quantistici basati su qubit superconduttori. Permette di scrivere programmi in Python utilizzando librerie specifiche per la definizione di circuiti quantistici.
- **Cirq**: una libreria Python sviluppata da Google per la programmazione di computer quantistici e simulazioni quantistiche.
- **Q#**: un linguaggio di programmazione sviluppato da Microsoft specificamente per la programmazione quantistica, integrato nell'ecosistema .NET.
- **Forest SDK**: sviluppato da Rigetti Computing, include il linguaggio Quil per la programmazione quantistica.
- **Pennylane**: una libreria per il machine learning quantistico che permette di interfacciarsi con diversi back-end quantistici.

### 8.5.5 Simulatori quantistici

Poiché l'accesso all'hardware quantistico reale è limitato, spesso si utilizzano simulatori quantistici su computer classici per sviluppare e testare programmi quantistici. Questi simulatori emulano il comportamento di un circuito quantistico, permettendo ai programmatori di verificare la correttezza dei loro algoritmi.

Tuttavia, è importante ricordare che i simulatori classici sono limitati dalla capacità computazionale classica e non possono scalare efficacemente per simulare sistemi quantistici di grandi dimensioni.

### 8.5.6 Conclusione

La programmazione quantistica rappresenta un'interfaccia tra il mondo classico e quello quantistico. Sebbene l'esecuzione dei calcoli avvenga nel dominio quantistico, la definizione e il controllo dei calcoli stessi avviene attraverso sistemi classici. Questa interazione tra hardware quantistico e software classico è fondamentale per l'avanzamento della tecnologia quantistica e per la realizzazione di applicazioni pratiche.

## 8.6 Bibliografia Ragionata

Per approfondire i temi trattati in questo capitolo, è fondamentale consultare una serie di opere e articoli che hanno contribuito in modo significativo allo sviluppo della computazione quantistica e alla comprensione delle sfide tecnologiche associate.

Nielsen and Chuang [2010], nel loro testo *Quantum Computation and Quantum Information*, offrono una trattazione completa dei principi fondamentali della computazione quantistica, compresa una discussione dettagliata dei criteri di DiVincenzo. Gli autori esplorano le diverse tecnologie per la realizzazione di qubit, le sfide della decoerenza e le strategie per la correzione degli errori quantistici.

DiVincenzo [2000], nell'articolo *The Physical Implementation of Quantum Computation*, presentano una panoramica dei requisiti fisici per la realizzazione di un computer quantistico, introducendo i sette criteri che portano il suo nome. Questo lavoro è fondamentale per comprendere le basi teoriche e pratiche necessarie per l'implementazione dell'elaborazione quantistica.

Nel libro *Quantum Optics*, Garrison and Chiao [2008] approfondiscono le proprietà dei fotoni e le tecniche dell'ottica quantistica, fornendo le conoscenze necessarie per comprendere come i fotoni possano essere utilizzati come qubit in un computer quantistico fotonico. Il testo esplora anche i fenomeni di decoerenza e le interazioni luce-materia.

O'Brien [2007], nell'articolo *Optical Quantum Computing*, discutono le prospettive e le sfide della computazione quantistica basata su fotoni, evidenziando le tecnologie attuali e le direzioni future della ricerca. Gli autori analizzano le implementazioni sperimentali dei gate quantistici ottici e le strategie per scalare i sistemi fotonici.

Per una comprensione più dettagliata dei fenomeni di decoerenza e delle tecniche per la correzione degli errori quantistici, il libro di Lidar and Brun [2013], *Quantum Error Correction*, è una risorsa inestimabile. Esso esplora i principi teorici e le applicazioni pratiche dei codici quantistici, fondamentali per mantenere l'integrità dell'informazione quantistica nei computer quantistici reali.

Infine, per uno sguardo sulle tecnologie emergenti e sulle sfide pratiche nella costruzione di computer quantistici, l'articolo di Arute et al. [2019], *Quantum Supremacy Using a Programmable Superconducting Processor*, presenta i risultati di Google nel raggiungimento della cosiddetta "supremazia quantistica" utilizzando un processore quantistico superconduttore. Questo lavoro illustra le capacità attuali dei computer quantistici e le sfide tecnologiche affrontate per raggiungere tali traguardi.

## 8.7 Esercizi pratici con QuantumSim

Per esplorare le interazioni tra componenti nei computer quantistici, utilizziamo **QuantumSim** per simulare un circuito che include qubit ancillari per la correzione degli errori e un semplice programma quantistico.

### 8.7.1 Simulazione di correzione degli errori

La correzione degli errori è fondamentale per mantenere l'informazione nei computer quantistici. Simuliamo il codice di ripetizione quantistico a tre qubit, che protegge uno stato quantistico da un singolo errore bit-flip.

```
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
    QubitState* state = initializeState(3); // Tre qubit: 1 dati, 2
    ancillari

    // Stato iniziale del qubit dati
    initializeStateTo(state, 0);
    printf("Stato iniziale:\n");
    printState(state);

    // Codifica ridondante (CNOT per la duplicazione)
    applyCNOT(state, 0, 1);
    applyCNOT(state, 0, 2);
    printf("Stato dopo la codifica:\n");
    printState(state);

    // Simula un errore bit-flip sul secondo qubit
    applyX(state, 1);
    printf("Stato dopo il bit-flip:\n");
    printState(state);
```

```
22
23      // Decodifica e correzione (CNOT inversi e misura)
24      applyCNOT(state, 0, 1);
25      applyCNOT(state, 0, 2);
26      MeasurementResult result1 = measure(state, 1);
27      MeasurementResult result2 = measure(state, 2);
28
29      if (result1.result != result2.result) {
30          applyX(state, 0); // Corregge l'errore
31      }
32
33      printf("Stato finale dopo la correzione:\n");
34      printState(state);
35
36      // Libera la memoria allocata
37      freeState(state);
38      return 0;
39  }
```

Listing 8.1: circuit8.1.c

**Spiegazione:**

- Il qubit dati viene duplicato su due qubit ancillari.
- Un errore bit-flip viene introdotto su uno dei qubit.
- La misura dei qubit ancillari rileva e corregge l'errore, ripristinando lo stato originale.

### 8.7.2 Simulazione di un semplice programma quantistico

Definiamo un programma che prepara un circuito quantistico, esegue operazioni su qubit e misura i risultati.

```
1  #include "quantum_sim.h"
2  #include <stdio.h>
3
4  int circuit(void) {
5      QubitState* state = initializeState(2); // Due qubit
6
7      // Stato iniziale
8      initializeStateTo(state, 0);
9      printf("Stato iniziale:\n");
10     printState(state);
11
12     // Applica gate Hadamard al primo qubit
13     applyHadamard(state, 0);
14     printf("Stato dopo Hadamard:\n");
15     printState(state);
16
17     // Applica gate CNOT
```

```
18      applyCNOT(state, 0, 1);
19      printf("Stato dopo CNOT:\n");
20      printState(state);
21
22      // Misura
23      MeasurementResult result1 = measure(state, 0);
24      MeasurementResult result2 = measure(state, 1);
25      printf("Risultati delle misure: %d, %d\n", result1.result, result2.
        result);
26
27      // Libera la memoria allocata
28      freeState(state);
29      return 0;
30  }
```

Listing 8.2: circuit8.2.c

**Spiegazione:**

- Il circuito prepara uno stato sovrapposto con il gate Hadamard e lo entangla con il CNOT.
- La misura restituisce risultati correlati grazie all'entanglement.

## Conclusione

Questi esercizi dimostrano come QuantumSim possa essere utilizzato per simulare componenti chiave dei computer quantistici, come la correzione degli errori e l'esecuzione di programmi semplici. Nel **Capitolo 9**, esploreremo ulteriormente algoritmi quantistici come l'algoritmo di Grover e Shor, che sfruttano appieno il potenziale della computazione quantistica.

# Capitolo 9

# Programmazione di un circuito quantistico

## 9.1 Introduzione

La computazione quantistica rappresenta una rivoluzione nel modo in cui affrontiamo problemi computazionali, offrendo potenziali vantaggi significativi rispetto ai computer classici in specifici ambiti. Nel capitolo precedente, abbiamo esplorato come la meccanica quantistica possa portare a fenomeni unici come la **non clonazione**, il **teletrasporto** e il **dense coding**, concetti senza analoghi nel mondo classico. Tuttavia, per sfruttare appieno il potenziale dei computer quantistici, é fondamentale comprendere come programmare e implementare algoritmi quantistici su circuiti quantistici reali.

In questo capitolo, ci concentreremo sulla **programmazione di un circuito quantistico**, illustrando come i **gate quantistici** possono essere utilizzati per costruire algoritmi che sfruttano i principi di *sovrapposizione* e *interferenza quantistica*. Attraverso l'esempio del **Problema di Deutsch**, mostreremo come un algoritmo quantistico possa risolvere un problema con un'efficienza impossibile da raggiungere con metodi classici.

Il Problema di Deutsch rappresenta uno dei primi algoritmi quantistici che dimostrano un vantaggio quantistico rispetto agli algoritmi classici. Il problema consiste nel determinare se una funzione booleana sconosciuta $f(x)$, che accetta in ingresso un singolo bit, sia **costante** (restituisce lo stesso valore per entrambi gli input) o **bilanciata** (restituisce valori diversi per input diversi). Mentre un algoritmo classico richiede due valutazioni della funzione per risolvere il problema nel caso peggiore, l'algoritmo quantistico di Deutsch può determinare la natura della funzione con una sola valutazione, sfruttando la sovrapposizione degli stati quantistici.

Nel corso del capitolo, illustreremo:

- **La progettazione del circuito quantistico**: descriveremo passo passo come costruire il circuito che implementa l'algoritmo di Deutsch, utilizzando i gate quantistici fondamentali come l'**operatore di Hadamard** e il **gate CNOT**.

- **L'implementazione in codice QASM**: presenteremo un esempio concreto di come programmare il circuito utilizzando il *Quantum Assembly Language* (QASM), un linguaggio standard per la descrizione di circuiti quantistici.
- **L'esecuzione e l'interpretazione dei risultati**: spiegheremo come eseguire il programma su un computer quantistico (reale o simulato) e come interpretare i risultati ottenuti dalla misura dei qubit.
- **Le implicazioni e le conclusioni**: discuteremo il significato del vantaggio quantistico dimostrato dall'algoritmo di Deutsch e le prospettive future della programmazione quantistica.

Questo capitolo fornisce un ponte tra la teoria e la pratica della computazione quantistica, mostrando come gli algoritmi quantistici possono essere effettivamente implementati e programmati. Comprendere questi aspetti pratici é fondamentale per chiunque voglia avvicinarsi al campo dell'informatica quantistica, sia dal punto di vista teorico che applicativo.

## 9.2 Soluzioni quantistiche a problemi classici

Nel capitolo 6 é stata sviluppata l'idea di *gate quantistici* e si é visto come questi possono essere usati per costruire circuiti reversibili analoghi a quelli classici, come ad esempio il circuito sommatore. Nel capitolo precedente si é utilizzata la meccanica quantistica per ottenere risultati come la non clonazione, il teletrasporto e la codifica densa, che non hanno un analogo classico.

Per apprezzare la differenza tra la computazione classica (o convenzionale) e quella quantistica, vediamo una semplice quanto efficace applicazione del principio di *superposizione* e *interferenza quantistica*. Il problema che stiamo per affrontare consiste nello stabilire se una certa funzione $f(x)$, di cui non si conosce l'implementazione, sia una funzione costante oppure bilanciata.

Per ridurre al minimo la complessitá della presentazione, facciamo l'ipotesi semplificativa che la funzione accetti un solo bit di informazione. In tal caso possiamo distinguere i due casi, cioé costante o bilanciata, come segue:

**Costante:** $f(0) = f(1) = c$, con $c \in \{0, 1\}$

**Bilanciata:** $f(0) \neq f(1)$

Se la funzione $f(x)$ é ignota, l'unica strategia classica con cui si può distinguere tra i due casi é quella di valutare la funzione due volte, passando la prima volta come argomento 0 e la seconda volta 1.

Per esempio, in linguaggio C avremmo:

```
// Funzione costante
int f(int x)
{
    return 0; // Restituisce sempre 0
}

// Funzione bilanciata
int f(int x)
{
    return x; // Restituisce x
}
```

Listing 9.1: Esempio di utilizzo di QuantumSim

Per testare i due casi si ha:

```
int r[2];
for(int i = 0; i <= 1; i++)
{
    r[i] = f(i);
}
if(r[0] == r[1])
{
    // La funzione \'e costante
}
else
{
    // La funzione \'e bilanciata
}
```

Listing 9.2: Esempio di utilizzo di QuantumSim

É chiaro che, anche utilizzando soluzioni alternative, il codice della funzione $f$ viene eseguito due volte. Usando un algoritmo quantistico, possiamo ridurre il numero di valutazioni necessarie ad una sola chiamata, risolvendo il problema dimezzando quella che viene chiamata *complessitá di query*. Il problema che stiamo per affrontare é noto come **Problema di Deutsch**.

## 9.3 Progettazione del circuito

L'obiettivo é mostrare che la computazione quantistica può risolvere il Problema di Deutsch usando una sola valutazione della funzione $f$. Per dimostrare questa affermazione, dobbiamo progettare un circuito che implementi l'algoritmo.

Per mantenere la reversibilitá della computazione, rappresentiamo la funzione $f(x)$ come un'operazione unitaria $U_f$ che agisce su due qubit:

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

dove $\oplus$ indica la somma modulo 2 (operazione XOR). Questo operatore é unitario e reversibile.

Il circuito quantistico si sviluppa nei seguenti passi:

1. Preparazione dello stato iniziale:

$$|\psi_0\rangle = |0\rangle \otimes |1\rangle$$

2. Applicazione dei gate di Hadamard ai due qubit:

$$|\psi_1\rangle = (H \otimes H)|\psi_0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

3. Applicazione dell'operatore $U_f$:

$$|\psi_2\rangle = U_f|\psi_1\rangle$$

4. Applicazione di una porta di Hadamard al primo qubit:

$$|\psi_3\rangle = (H \otimes I)|\psi_2\rangle$$

5. Misura del primo qubit.

Dopo i calcoli, si trova che:
- Se $f$ é costante, il primo qubit sará nello stato $|0\rangle$ con probabilitá 1. - Se $f$ é bilanciata, il primo qubit sará nello stato $|1\rangle$ con probabilitá 1.

Quindi, con una sola valutazione di $f$, possiamo determinare se la funzione é costante o bilanciata.

## 9.4 Codice QASM

Per implementare questo algoritmo su un computer quantistico reale o simulato, possiamo utilizzare il linguaggio QASM (Quantum Assembly Language). Ecco un esempio di codice:

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[1];

// Inizializzazione
x q[1];       // Applica X al secondo qubit per ottenere |1>

// Applica Hadamard ai due qubit
h q[0];
h q[1];
```

```
13
14 // Definizione della funzione U_f
15 // Scegliere l'implementazione di U_f in base alla funzione f
16
17 // Caso costante: f(x) = 0
18 // U_f non fa nulla (identit\'a)
19 // // Nessuna operazione
20
21 // Caso costante: f(x) = 1
22 // U_f = Z gate sul secondo qubit
23 // z q[1];
24
25 // Caso bilanciata: f(x) = x
26 // U_f = CNOT gate con controllo su q[0] e target su q[1]
27 cx q[0], q[1];
28
29 // Caso bilanciata: f(x) = NOT x
30 // U_f = CNOT seguito da X sul secondo qubit
31 // cx q[0], q[1];
32 // x q[1];
33
34 // Applica Hadamard al primo qubit
35 h q[0];
36
37 // Misura il primo qubit
38 measure q[0] -> c[0];
```

Listing 9.3: Esempio di utilizzo di QuantumSim

Per testare i diversi casi, si può modificare la sezione del codice relativa all'operazione $U_f$:

- Per $f(x) = 0$ (costante), non si applica alcuna operazione. - Per $f(x) = 1$ (costante), si applica una porta $Z$ sul secondo qubit. - Per $f(x) = x$ (bilanciata), si applica una porta CNOT. - Per $f(x) = \overline{x}$ (bilanciata), si applica una porta CNOT seguita da una porta $X$ sul secondo qubit.

## 9.5 Esecuzione del programma

L'esecuzione del programma termina con la misura del qubit $q[0]$, il cui risultato viene memorizzato nel registro classico $c[0]$. Valutando il valore di $c[0]$, possiamo determinare il risultato della computazione quantistica:

- Se $c[0] = 0$, la funzione $f$ é **costante**. - Se $c[0] = 1$, la funzione $f$ é **bilanciata**.

Per provare il codice e testare le diverse eventualitá, si modificano le operazioni che implementano $U_f$ come descritto nella sezione precedente.

ai risultati corretti e diminuire le ampieste relative ai risultati sbagliati ha un costo minore.

## 9.6 I segreti della computazione quantistica

Abbiamo visto come l'algoritmo di Deutsch sfrutta la sovrapposizione quantistica per determinare in modo deterministico se una funzione sia bilanciata o non bilanciata. Quella di Deutsch é un caso quasi unico di una soluzione quantistica deterministica a un problema. Tuttavia, la vera potenza del calcolo quantistico si esprime principalmente attraverso la sua natura probabilistica. Sebbene ciò, inizialmente, possa sembrare controintuitivo per un neofita, l'esperienza dimostra che i vantaggi di un risultato probabilistico superano quelli di un risultato deterministico quando la complessitá computazionale richiesta per quest'ultimo é molto più alta.

In pratica, il segreto del calcolo quantistico consiste nel rappresentare le varie possibili soluzioni di un problema come stati quantistici, ognuno con una propria ampiezza di probabilitá. Successivamente, viene eseguita una trasformazione specifica che riduce le ampiezze di probabilitá degli stati non corretti e aumenta quelle degli stati corretti. L'algoritmo di Grover, che vedremo di seguito, rappresenta un esempio emblematico di come la ricerca di un dato noto solo all'oracolo possa avvenire amplificando l'ampiezza di probabilitá relativa alla soluzione esatta.

Questa metodologia potrebbe sembrare fantascientifica o prossima alla fantasia. Ciò accade spesso perché si confonde la difficoltá intrinseca della meccanica quantistica con la metodologia innovativa del calcolo. Per chiarire questo aspetto, propongo un esempio classico che ci guiderá nella comprensione dell'approccio quantistico. Sebbene l'esempio classico abbia dei limiti, poiché utilizza la fisica classica, offre comunque un'intuizione fondamentale per comprendere come sia possibile ottenere un risultato corretto attraverso un approccio probabilistico. Per illustrare questo concetto, sfrutteremo l'idea dei solidi platonici, ovvero solidi con tutte le facce costituite da poligoni regolari uguali.

### 9.6.1 La natura probabilistica del calcolo quantistico

Consideriamo un icosaedro, un solido regolare con 20 facce triangolari equilateri. Se l'icosaedro viene lanciato come un dado, ogni faccia ha la stessa probabilitá di risultare vincente. Numerando le facce da 0 a 19, ogni faccia ha una probabilitá su 20 di uscire. Ad esempio, su 200 lanci ci si aspetta che, approssimativamente, ogni faccia esca circa 10 volte.

Immaginiamo ora che l'icosaedro sia "truccato", ovvero che la massa non sia distribuita in modo uniforme, ma che il centro di massa si trovi sotto una faccia specifica. In questo caso, lanciando il dado, la probabilitá che la faccia opposta al centro di massa esca aumenta significativamente. Questo principio può essere sfruttato per costruire una calcolatrice particolare.

Supponiamo che toccando le facce dell'icosaedro sia possibile impostare due addendi: uno compreso tra 0 e 9 e l'altro tra 0 e 10. Dopo aver toccato le due facce corrispondenti agli addendi, un sistema meccanico interno sposta il centro di massa sotto la faccia corrispondente alla loro somma. Ad esempio, impostando i numeri 2 e 3, il sistema sposta il

centro di massa sotto la faccia opposta al numero 5. Quando l'icosaedro viene lanciato, la faccia con il numero 5 ha la probabilitá maggiore di risultare vincente.

Questo meccanismo, tuttavia, non é deterministico. Il sistema aumenta la probabilitá che il numero corretto emerga, ma il risultato resta probabilistico. Questo concetto fondamentale é analogo a ciò che accade nel calcolo quantistico, dove l'amplificazione delle ampiezze relative ai risultati corretti riduce quelle degli stati sbagliati.

### 9.6.2 Differenze tra il classico e il quantistico

Perché non utilizziamo un approccio simile nel contesto classico? La risposta sta nel costo computazionale. Nel mondo classico, spostare oggettivamente il centro di massa sotto la faccia corrispondente alla somma richiede una complessitá computazionale simile a quella necessaria per calcolare direttamente il risultato. Nel calcolo quantistico, invece, amplificare le ampiezze relative ai risultati corretti e ridurre quelle relative ai risultati sbagliati può avvenire con un costo computazionale significativamente inferiore.

### 9.6.3 L'algoritmo di Grover: ricerca non strutturata e il ruolo dell'oracolo

L'algoritmo di Grover é progettato per risolvere il problema della ricerca non strutturata, in cui l'obiettivo é identificare un elemento specifico all'interno di uno spazio di possibilitá, senza alcuna struttura evidente che guidi la ricerca. Prima di approfondire l'algoritmo, é importante comprendere la differenza tra database strutturati e non strutturati.

#### Database strutturati vs. non strutturati

Un database strutturato é organizzato in modo che gli elementi siano disposti secondo una logica precisa, spesso rappresentata attraverso tabelle, indici o grafi. Ad esempio, un elenco telefonico é strutturato in modo che i nomi siano ordinati alfabeticamente, facilitando la ricerca di un numero specifico associato a un nome.

Un database non strutturato, invece, non presenta alcuna logica o schema organizzativo evidente. Gli elementi sono semplicemente distribuiti, senza un ordine particolare. Ad esempio, un mazzo di carte mescolato rappresenta un database non strutturato: per trovare un asso di cuori, é necessario controllare ogni carta, una alla volta, fino a trovarlo.

L'algoritmo di Grover é rivoluzionario perché riduce a $O(\sqrt{N})$ il tempo di ricerca in database non strutturati, dove i metodi classici richiederebbero un numero di tentativi proporzionale al numero totale di elementi ($O(N)$). Cioé un algortmo classico richiede un numero di passi proporzionale (circa la metá) al numero di "oggetti" da verificare, quello classico é invece proporzionale alla radice quadrata di tale numero.

### Il ruolo dell'oracolo

Il cuore dell'algoritmo di Grover é l'oracolo, una funzione quantistica che, dato uno stato $|x\rangle$, indica se $x$ é la soluzione corretta.

Un'obiezione comune é: "Se l'oracolo conosce la risposta, allora so giá la risposta anch'io." Questa interpretazione, però, é errata. L'oracolo non é una conoscenza esplicita propria dell'essere umano. L'oracolo è una struttura fisica che agisce da "filtro" per il sistema quantistico. Ad esempio, l'oracolo potrebbe essere rappresentato da una proteina che si lega solo a un determinato sistema chimico. Non conosciamo a priori quale sistema causi il legame, ma l'oracolo risponde fisicamente quando gli viene presentato il sistema corretto.

### Esempio pratico: ricerca di un substrato chimico

Immaginiamo di avere un laboratorio in cui vogliamo identificare quale molecola, tra $N$ molecole diverse, si lega a una proteina specifica. La proteina agisce come un oracolo: é in grado di riconoscere la molecola corretta e innescare una reazione. Tuttavia, non possiamo conoscere la molecola corretta senza provare ogni combinazione.

Utilizzando l'algoritmo di Grover, rappresentiamo tutte le molecole come stati quantistici. L'oracolo agisce invertendo il segno dell'ampiezza associata allo stato corretto (la molecola che si lega). Attraverso il meccanismo di amplificazione delle ampiezze, lo stato corrispondente alla molecola corretta emerge con una probabilitá significativamente maggiore rispetto agli altri, permettendoci di identificarlo con una misura.

## Come funziona l'algoritmo di Grover

Consideriamo un sistema di $n$ qubit che possono trovarsi in $N = 2^n$ possibili stati. Indichiamo con $x \in \{0, 1, 2..., N-1\}$ gli stati del sistema. L'azione dell'oracolo è formalizzata attraverso la funzione $f(x)$ che vale 1 per gli stati *target*, o marcati, e 0 per gli altri stati (stati di *default*). Definiamo l'isieme $T = \{x|f(x) = 1\}$ come l'insieme degli stati target e $D = \{x|f(x) = 0\}$ come l'insieme degli stati di default. Il numero degli stati nei due insiemi è espresso attraverso l'operatore di cardinalità, cioè $|T|$ e $|D|$.

Con queste prime formalizzazioni, possiamo scrivere l'algoritmo di Grover:

1. Inizializzazione: il sistema é preparato in una sovrapposizione uniforme di tutti gli stati possibili:
$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

2. Applicazione dell'oracolo: l'oracolo inverte il segno dell'ampiezza associata allo stato marcato (stato target).

3. Operatore di diffusione: l'operatore di Grover riflette lo stato del sistema rispetto alla media degli stati, amplificando di conseguenza l'ampiezza dello stato corretto che a causa della inversione di segno si trova giocoforza più lontano dalla media.

4. Iterazione: i passi 2 e 3 sono ripetuti un numero ottimale di volte che calcoleremo a breve.

5. Misura: lo stato finale, con alta probabilitá, corrisponde alla soluzione corretta.

In sintesi, l'oracolo non é una conoscenza esplicita, ma un elemento fisico o logico che guida l'amplificazione delle ampiezze nel sistema quantistico. L'algoritmo di Grover sfrutta questa guida per trasformare un problema di ricerca non strutturata in un processo computazionale quantistico altamente efficiente.

### Oracolo $U$

L'oracolo $U$ è definito come un operatore unitario che agisce su due registri: il registro dati $|x\rangle$ e il registro ausiliario $|a\rangle$. Matematicamente, è descritto come:

$$U|x\rangle|a\rangle = |x\rangle|f(x) \oplus a\rangle,$$

dove:

- $f(x)$ è la funzione booleana, definita sopra, che indica se lo stato $x$ è uno stato target ($f(x) = 1$) oppure uno stato non target ($f(x) = 0$).
- $\oplus$ rappresenta l'operazione XOR.

Se inizializziamo il registro ausiliario nello stato:

$$|a\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

l'operatore $U$ agisce come segue:

$$U|x\rangle|-\rangle = \begin{cases} -|x\rangle|-\rangle, & \text{se } f(x) = 1, \\ |x\rangle|-\rangle, & \text{se } f(x) = 0. \end{cases}$$

Quindi, l'oracolo identifica gli stati target (f(x)=1) e inverte il segno della loro ampiezza. In pratica, una volta dimostrato che l'operatore $U$ inverte il segno degli stati target, si può *trascurare* il registro ausiliario $a$ e implementare direttamente l'operatore $U$ come appena descritto. Questo ragionamento è valido quando l'operatore $U$ è costruito tramite gate a scopo dimostrativo; tuttavia, in scenari pratici, non è sempre garantito che il registro ausiliario possa essere ignorato.

### Separazione logica degli stati

Nell'algoritmo di Grover, lo spazio degli stati quantistici è suddiviso in due insiemi distinti:

- **Stati target ($T$):** Gli stati che vogliamo identificare tramite l'algoritmo. Questi stati sono "marcati" dall'oracolo e rappresentano le soluzioni al problema. Indichiamo la sovrapposizione uniforme degli stati target con $|\psi_T\rangle$, definita come:

$$|\psi_T\rangle = \frac{1}{\sqrt{|T|}} \sum_{x \in T} |x\rangle,$$

  dove $|T|$ è il numero di stati target.

- **Stati di default ($D$):** Tutti gli altri stati che non sono target. Questi stati non sono marcati dall'oracolo. Indichiamo la sovrapposizione uniforme degli stati di default con $|\psi_D\rangle$, definita come:

$$|\psi_D\rangle = \frac{1}{\sqrt{|D|}} \sum_{x \in D} |x\rangle,$$

  dove $|D|$ è il numero di stati di default e $|D| + |T| = N$, con $N = 2^n$ che, come visto poco sopram rappresenta il numero totale di stati per il sistema a $n$ qubit.

Questa distinzione tra stati target e stati di default è essenziale per comprendere il funzionamento dell'oracolo e dell'operatore di diffusione nell'algoritmo di Grover. L'oracolo infatti modifica gli stati target invertendo il segno delle loro ampiezze, mentre l'operatore di diffusione amplifica queste ampiezze per renderle maggiori di quelle degli stati di default.

Consideriamo lo stato iniziale (indice 0) $|\psi_0\rangle$ del sistema prima di agire su di esso con l'algoritmo di Grover. Tale stato può essere rappresentato come una combinazione lineare di $|\psi_T\rangle$ e $|\psi_D\rangle$:

$$|\psi\rangle = t_0|\psi_T\rangle + d_0|\psi_D\rangle,$$

dove:

- $t_0$ è il coefficiente associato agli stati target e rappresenta il contributo degli stati target nella sovrapposizione.

- $d_0$ è il coefficiente associato agli stati di default e rappresenta il contributo degli stati di default nella sovrapposizione.

L'idea fondamentale dell'algoritmo di Grover è quella di amplificare progressivamente le ampiezze degli stati target $|\psi_T\rangle$ rispetto agli stati di default $|\psi_D\rangle$ attraverso l'applicazione iterativa di un operatore $O$. Questo operatore realizza la seguente trasformazione:

$$O\big(t_i|\psi_T\rangle + d_i|\psi_D\rangle\big) = t_{i+1}|\psi_T\rangle + d_{i+1}|\psi_D\rangle,$$

dove:

- $t_i$ è l'ampiezza degli stati target ($|\psi_T\rangle$) alla $i$-esima iterazione.

- $d_i$ è il contributo degli stati di default ($|\psi_D\rangle$) alla $i$-esima iterazione.
- Dopo ogni iterazione, $t_{i+1} > t_i$, cioè il contributo degli stati target aumenta.
- Analogamente, $d_{i+1} < d_i$, cioè il contributo degli stati di default diminuisce.

### Meccanismo di Amplificazione

Questa trasformazione operata da $O$ è il risultato combinato dell'applicazione dell'oracolo $U$ e dell'operatore di diffusione $D$:

1. L'oracolo $U$ inverte il segno delle ampiezze degli stati target $|\psi_T\rangle$, creando una differenza più marcata tra $t_i$ e $d_i$.
2. L'operatore di diffusione $D$ esegue una riflessione rispetto alla media delle ampiezze, amplificando $t_i$ e riducendo $d_i$.

Attraverso un numero ottimale di iterazioni, pari a:

$$R = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{|T|}} \right\rfloor,$$

l'algoritmo garantisce che le ampiezze degli stati target $t_R$ diventino predominanti, massimizzando la probabilità di misurare uno stato target $|\psi_T\rangle$. Nel contempo, le ampiezze degli stati di default $d_R$ diventano trascurabili.

L'operatore di diffusione è responsabile dell'amplificazione delle ampiezze degli stati target ($T$) rispetto agli stati di default ($D$).

Esso è definito attraverso L'operatore Hadamard-Walsh ($H^{\otimes n}$) che è l'analogo dell'operatore $H$ ma definito su stati per $n$ qubit e l'operatore $S_0^\pi$ che inverte il segno del qubit di indice 0. La sua definizione formale è:

$$D = -WS_0^\pi W$$

L'operatore $O$ può quindi essere scritto come:

$$O = -WS_0^\pi WU$$

dove $U$ è l'oracolo.

### Iterazioni Successive

Ad ogni iterazione, l'operatore $O$ continua a:

1. Amplificare le ampiezze degli stati target $T$.
2. Ridurre le ampiezze degli stati di default $D$.

Il processo si ripete fino a raggiungere un numero ottimale di iterazioni:

$$R = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{|T|}} \right\rfloor,$$

## 9.7 Conclusioni

Con questo capitolo si é presentato un esempio concreto di algoritmo quantistico che mostra un vantaggio rispetto alla computazione classica. Il **Problema di Deutsch** evidenzia come la meccanica quantistica possa essere sfruttata per ridurre la complessitá di query di certi problemi.

La computazione quantistica ha il potenziale di risolvere problemi in modo più efficiente rispetto ai metodi classici per specifici casi. Tuttavia, non si prevede che la computazione quantistica sostituirá completamente quella classica. Piuttosto, le due tecnologie coesisteranno, con la computazione quantistica che affiancherá quella classica per affrontare problemi dove può offrire vantaggi significativi.

## 9.8 Bibliografia Ragionata

Per approfondire i concetti presentati in questo capitolo e ottenere una comprensione più completa della programmazione di circuiti quantistici, é consigliabile consultare le seguenti opere chiave nel campo dell'informatica quantistica.

Nielsen and Chuang [2010], nel loro testo *Quantum Computation and Quantum Information*, offrono una trattazione esaustiva dei principi fondamentali dell'informatica quantistica. Il libro include una descrizione dettagliata dell'algoritmo di Deutsch e delle sue generalizzazioni, fornendo sia l'intuizione fisica che l'analisi matematica rigorosa. Gli autori discutono anche i linguaggi di programmazione quantistica e gli strumenti utilizzati per la simulazione e l'implementazione degli algoritmi.

L'articolo originale di Deutsch [1985], *Quantum theory, the Church–Turing principle and the universal quantum computer*, introduce il concetto di algoritmo quantistico e presenta il Problema di Deutsch come primo esempio di un problema in cui un computer quantistico può superare un computer classico. Questo lavoro pionieristico ha aperto la strada alla ricerca successiva nel campo degli algoritmi quantistici.

Deutsch and Jozsa [1992], nell'articolo *Rapid solution of problems by quantum computation*, estendono l'algoritmo originale introducendo il **Problema di Deutsch-Jozsa**, che generalizza il problema a funzioni con $n$ bit in ingresso. Questo articolo é fondamentale per comprendere come i principi utilizzati nel Problema di Deutsch possono essere applicati a problemi più complessi.

Per una prospettiva più pratica sulla programmazione quantistica, il libro di Benenti et al. [2004], *Principles of Quantum Computation and Information*, offre una panoramica delle tecniche di implementazione degli algoritmi quantistici, inclusa la descrizione di

linguaggi di programmazione e ambienti di simulazione. Gli autori forniscono esempi di codice e discutono le sfide associate alla realizzazione pratica dei circuiti quantistici.

Gay and Mackie [2009], nel libro *Semantic techniques in quantum computation*, esplorano i linguaggi di programmazione sviluppati specificamente per la computazione quantistica. Il testo discute le caratteristiche dei linguaggi quantistici, come QASM, e illustra come questi possono essere utilizzati per programmare algoritmi quantistici complessi.

Infine, per chi é interessato agli aspetti sperimentali e all'implementazione su hardware reale, l'articolo di Barends et al. [2014], *Superconducting quantum circuits at the surface code threshold for fault tolerance*, descrive come gli algoritmi quantistici possono essere eseguiti su dispositivi quantistici superconduttori. Questo lavoro evidenzia le sfide e i progressi nella realizzazione di computer quantistici pratici.

## 9.9 Esercizi pratici con QuantumSim

Utilizziamo **QuantumSim** per simulare il circuito che risolve il **Problema di Deutsch**, testando sia il caso costante che il caso bilanciato e poi l'algoritmo di grover per 2 e 3 qubit.

### 9.9.1 Simulazione del Problema di Deutsch

Il Problema di Deutsch permette di determinare se una funzione booleana $f(x)$ é costante o bilanciata con una sola valutazione. Implementiamo il circuito quantistico utilizzando QuantumSim.

```
#include "quantum_sim.h"
#include <stdio.h>

// Definizione della funzione f(x)
void applyUf(QubitState* state, int x, int y, int type) {
    switch (type) {
        case 0: // f(x) = 0, costante
            // Nessuna operazione
            break;
        case 1: // f(x) = 1, costante
            applyZ(state, y);
            break;
        case 2: // f(x) = x, bilanciata
            applyCNOT(state, x, y);
            break;
        case 3: // f(x) = !x, bilanciata
            applyCNOT(state, x, y);
            applyX(state, y);
            break;
    }
}

```

```c
int circuit(void) {
    QubitState* state = initializeState(2); // Due qubit: controllo e target

    // Stato iniziale |0>|1>

    applyX(state, 1);           // Configura il secondo qubit a |1>
    printf("Stato iniziale:\n");
    printState(state);

    // Applica Hadamard a entrambi i qubit
    applyHadamard(state, 0);
    applyHadamard(state, 1);
    printf("Stato dopo Hadamard:\n");
    printState(state);

    // Applicazione di U_f
    int type = 0; // Scegliere il tipo di funzione: 0, 1, 2 o 3
    applyUf(state, 0, 1, type);
    printf("Stato dopo U_f:\n");
    printState(state);

    // Applica Hadamard al primo qubit
    applyHadamard(state, 0);
    printf("Stato dopo Hadamard finale:\n");
    printState(state);

    // Misura del primo qubit
    MeasurementResult result = measure(state, 0);
    printf("Risultato della misura: %d\n", result.result);

    // Interpretazione del risultato
    if (result.result == 0) {
        printf("La funzione f(x) \'e costante.\n");
    } else {
        printf("La funzione f(x) \'e bilanciata.\n");
    }

    // Libera la memoria allocata
    freeState(state);
    return 0;
}
```

Listing 9.4: circuit9.1.c

**Spiegazione:**

- Lo stato iniziale é $|0\rangle|1\rangle$.
- Il gate di Hadamard crea uno stato di sovrapposizione.
- $U_f$ implementa la funzione $f(x)$ come operazione quantistica.

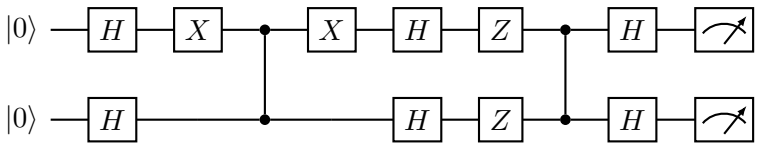
- Una secona porta Hadamard rendende possibile determinare se $f$ é costante o bilanciata.
- La misura sul primo qubit determina il risultato.

## Conclusione

Questo esercizio dimostra come usare QuantumSim per simulare un esempio fondamentale di algoritmo quantistico.

### 9.9.2 Simulazione dell'Algoritmo di Grover per 2 Qubit

L'algoritmo di Grover per 2 qubit consente di trovare uno stato target $|10\rangle$ in un database non strutturato contenente 4 stati con una sola iterazione dell'oracolo e dell'operatore di diffusione. Implementiamo il circuito utilizzando QuantumSim.




```c
#include "../../src/quantum_sim.h"
#include <math.h>
#include <stdio.h>

// Funzione per applicare Hadamard-Walsh
void applyHadamardWalsh(QubitState *state) {
    for (int i = 0; i < state->numQubits; i++) {
        applyHadamard(state, i);
    }
    printf("Applicato Hadamard-Walsh su tutti i qubit\n");
    printState(state);
}

// Funzione per applicare l'oracolo
void applyOracle(QubitState *state) {
    printf("Applicazione dell'oracolo\n");

    // Marca lo stato 10 usando un gate CZ
    applyX(state,0);
    applyCZ(state, 1, 0);
    applyX(state,0);
    printf(" Marcato lo stato 10\n");
      printState(state);
}

/**
 * Applica l'operatore di diffusione basato su Z e CZ
```

```
 */
void applyDiffusion(QubitState *state) {
    printf("Applicazione dell'operatore di diffusione\n");

    // Step 1: Hadamard su tutti i qubit
    for (int i = 0; i < state->numQubits; i++) {
        applyHadamard(state, i);
    }

    // Step 2: Z su tutti i qubit
    for (int i = 0; i < state->numQubits; i++) {
        applyZ(state, i); // Inverte il segno di tutti gli stati
    }

    // Step 3: Controlled-Z (CZ) tra i qubit
    applyCZ(state, 0, 1); // CZ applica una riflessione rispetto a |00>

    // Step 4: Hadamard su tutti i qubit
    for (int i = 0; i < state->numQubits; i++) {
        applyHadamard(state, i);
    }

    printf("Operatore di diffusione applicato\n");
    printState(state);
}


// Funzione principale
void circuit() {
    printf("Avvio dell'algoritmo di Grover\n");

    // Inizializza lo stato con 2 qubit
    QubitState *state = initializeState(2);
    printf("Stato iniziale con 2 qubit\n");
    printState(state);

    // Applica Hadamard-Walsh per creare la sovrapposizione uniforme
    applyHadamardWalsh(state);

    // Applica l'oracolo per marcare gli stati |11    e |01
    applyOracle(state);

    // Applica l'operatore di diffusione
    applyDiffusion(state);

    // Misura i qubit
    printf("Misurazione dello stato finale\n");
    for (int i = 0; i < state->numQubits; i++) {
        MeasurementResult result = measure(state, i);
        printf("Misura del qubit %d: %d\n", i, result.result);
```

```
78      }
79
80      printf("Stato finale dopo la misurazione\n");
81      printState(state);
82  }
```

Listing 9.5: circuit9.2.c

**Spiegazione:**

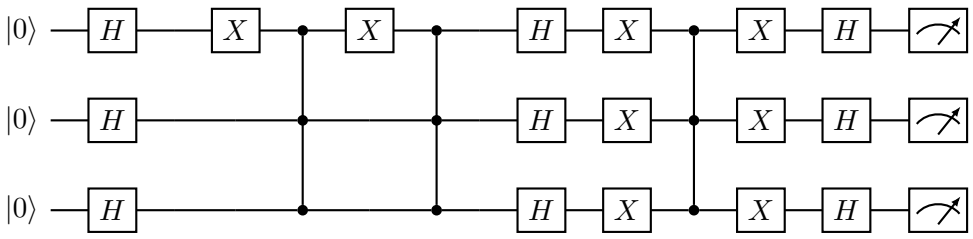
- Lo stato iniziale è $|00\rangle$.
- I gate di Hadamard-Walash ($H^{\otimes n}$) creano una sovrapposizione uniforme su tutti gli stati.
- L'oracolo utilizza i gate $X$ e $CZ$ per invertire il segno dello stato target $|10\rangle$.
- L'operatore di diffusione riflette gli stati rispetto alla media, amplificando l'ampiezza dello stato target.
- Una misura finale dei qubit permette di identificare lo stato target con alta probabilità.

## Conclusione

Questo esercizio mostra come l'algoritmo di Grover può essere utilizzato per cercare uno stato target in uno spazio di stati con 2 qubit. Il caso a 3 qubit segue lo stesso principio, ma con uno spazio di ricerca più ampio.

### 9.9.3 Simulazione dell'Algoritmo di Grover per 3 Qubit

L'algoritmo di Grover per 3 qubit permette di trovare due stati target $|110\rangle$ e $|111\rangle$ in un database non strutturato contenente 8 stati. Implementiamo il circuito utilizzando QuantumSim.




```
1  #include "../../src/quantum_sim.h"
2  #include <math.h>
3  #include <stdio.h>
4
5  #include "../../src/quantum_sim.h"
6  #include <math.h>
```

```c
#include <stdio.h>

// Funzione per applicare Hadamard-Walsh su tutti i qubit
void applyHadamardWalsh(QubitState *state) {
    for (int i = 0; i < state->numQubits; i++) {
        applyHadamard(state, i);
    }
    printf(" Applicato Hadamard-Walsh su tutti i qubit per creare la sovrapposizione uniforme\n");
    printState(state);
}

// Funzione per applicare l'oracolo
void applyOracle(QubitState *state) {
    printf(" Applicazione dell'oracolo per marcare gli stati target\n");

    // Marca lo stato |110    e |111    usando i gate CCZ
    applyX(state, 0);
    applyCCZ(state, 1, 2, 0);
    applyX(state, 0);
    applyCCZ(state, 1, 2, 0);

    printf("Stato marcato: |110    e |111    \n");
    printState(state);
}

/**
 * Applica l'operatore di diffusione basato su Z e CZ
 */
void applyDiffusion(QubitState *state) {
    printf(" Applicazione dell'operatore di diffusione per amplificare gli stati target\n");

    // Step 1: Applica Hadamard su tutti i qubit
    for (int i = 0; i < state->numQubits; i++) {
        applyHadamard(state, i);
    }

    // Step 2: Inverte il segno di tutti gli stati (applica gate X)
    for (int i = 0; i < state->numQubits; i++) {
        applyX(state, i);
    }

    // Step 3: Applica un gate CCZ per la riflessione rispetto a |000
    applyCCZ(state, 1, 2, 0);

    // Step 4: Ripristina i qubit applicando di nuovo X
    for (int i = 0; i < state->numQubits; i++) {
        applyX(state, i);
    }
```

```
55
56      // Step 5: Applica Hadamard su tutti i qubit per completare la
        riflessione
57      for (int i = 0; i < state->numQubits; i++) {
58          applyHadamard(state, i);
59      }
60
61      printf("Operatore di diffusione applicato con successo\n");
62      printState(state);
63  }
64
65  // Funzione principale per l'esecuzione dell'algoritmo di Grover
66  void circuit() {
67      printf(" Avvio dell'algoritmo di Grover\n");
68
69      // Inizializza lo stato con 3 qubit
70      QubitState *state = initializeState(3);
71      printf(" Stato iniziale con 3 qubit (|000   )\n");
72      printState(state);
73
74      // Step 1: Applica Hadamard-Walsh per creare la sovrapposizione
        uniforme
75      applyHadamardWalsh(state);
76
77      // Step 2: Applica l'oracolo per marcare gli stati target
78      applyOracle(state);
79
80      // Step 3: Applica l'operatore di diffusione per amplificare gli stati
        target
81      applyDiffusion(state);
82
83      // Misura i qubit
84      printf(" Misurazione dello stato finale\n");
85      for (int i = 0; i < state->numQubits; i++) {
86          MeasurementResult result = measure(state, i);
87          printf("Misura del qubit %d: %d\n", i, result.result);
88      }
89
90      printf(" Stato finale dopo la misurazione\n");
91      printState(state);
92  }
```

Listing 9.6: circuit9.3.c

**Spiegazione:**

- Lo stato iniziale è $|000\rangle$.
- Le porte Hadamard ($H^{\otimes n}$) creano una sovrapposizione uniforme su tutti gli stati.
- L'oracolo utilizza i gate $X$ e $CCZ$ per invertire il segno degli stati target $|110\rangle$ e $|111\rangle$.

- L'operatore di diffusione riflette gli stati rispetto alla media, amplificando le ampiezze degli stati target.
- Una misura finale dei qubit permette di identificare uno degli stati target con alta probabilità.

## Conclusione

Questo esercizio estende il caso a 2 qubit, mostrando come l'algoritmo di Grover scala con l'aumento dello spazio di ricerca. I principi rimangono gli stessi, ma la complessità del circuito aumenta con il numero di qubit.

# Appendice A

# Applicazione Pratica con QuantumSim

## A.1 Introduzione al Simulatore Quantistico

In questo capitolo, metteremo in pratica i concetti di informatica quantistica appresi finora utilizzando un simulatore quantistico scritto in linguaggio C. Questo simulatore ci permetterà di implementare e testare circuiti quantistici, fornendo una comprensione più profonda dei meccanismi alla base del calcolo quantistico.

### A.1.1 Perché Utilizzare un Simulatore?

L'uso di un simulatore quantistico offre numerosi vantaggi:

- **Apprendimento Pratico**: Sperimentare con un simulatore aiuta a consolidare la comprensione teorica attraverso l'implementazione concreta.
- **Accessibilità**: Poiché i computer quantistici reali sono ancora limitati e non facilmente accessibili, un simulatore é un'alternativa pratica per l'apprendimento e la ricerca.
- **Flessibilità**: Possiamo testare e modificare rapidamente circuiti quantistici di varia complessità, esplorando diversi algoritmi e fenomeni quantistici.

### A.1.2 Ambiente di Sviluppo

Per seguire questo capitolo, avrai bisogno di:

- Un compilatore C compatibile (ad esempio, GCC o Clang).
- Una conoscenza di base della programmazione in C.
- Il codice del simulatore quantistico scaricabile dal mio repository: *https://github.com/francescosisini/QuantumSim*

Per compilare il simulatore, puoi seguire direttamente le istruzioni che trovi su GitHub.

## A.2 Concetti Fondamentali Rivisitati

Prima di immergerci nel codice, rivediamo brevemente alcuni concetti chiave che saranno fondamentali per comprendere le implementazioni pratiche.

### A.2.1 Qubit e Stati Quantistici

Un **qubit** é l'unità fondamentale di informazione nel calcolo quantistico. A differenza di un bit classico, che può essere solo 0 o 1, un qubit può essere in una **sovrapposizione** di entrambi gli stati:

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

dove $\alpha$ e $\beta$ sono numeri complessi tali che $|\alpha|^2 + |\beta|^2 = 1$.

### A.2.2 Gate Quantistici

I **gate quantistici** sono operazioni realizzate tramite operatori unitari che manipolano lo stato dei qubit. Possono essere rappresentati per mezzo di matrici unitarie che agiscono sullo spazio degli stati dei qubit.

Alcuni dei gate fondamentali sono:

- **Pauli-X** ($X$): Analogo quantistico del NOT classico. Scambia gli stati $|0\rangle$ e $|1\rangle$.
- **Pauli-Y** ($Y$) e **Pauli-Z** ($Z$): Altri operatori di Pauli che introducono fasi specifiche.
- **Hadamard** ($H$): Crea una sovrapposizione equa tra $|0\rangle$ e $|1\rangle$.
- **Gate di Fase** ($S$, $T$): Introducono una fase specifica allo stato del qubit.
- **CNOT**: Un gate a due qubit che inverte il qubit target se il qubit di controllo é $|1\rangle$.

## A.3 Architettura del QuantumSim

In questa sezione, approfondiremo l'architettura del **QuantumSim**, analizzando le sue componenti principali e come interagiscono tra loro per simulare un sistema quantistico.

### A.3.1 Struttura del Codice

Il **QuantumSim** é organizzato in modo modulare, con funzioni specifiche per diverse operazioni quantistiche. Le componenti principali sono:

- **Rappresentazione dello Stato Quantistico**: Utilizza una struttura dati per memorizzare le ampiezze degli stati quantistici.

- **Inizializzazione dello Stato**: Funzioni per impostare lo stato iniziale del sistema quantistico.
- **Applicazione dei Gate Quantistici**: Funzioni che implementano i vari gate quantistici, modificando lo stato del sistema.
- **Misurazione**: Funzioni per misurare i qubit e collassare lo stato quantistico.
- **Gestione della Memoria**: Funzioni per allocare e liberare la memoria utilizzata dal simulatore.

### A.3.2 Rappresentazione dello Stato Quantistico

Lo stato quantistico é rappresentato dalla struttura `QubitState`:

```
typedef struct {
    int numQubits;
    double complex *amplitudes;
} QubitState;
```

Listing A.1: Definizione della struttura `QubitState`

- `numQubits`: Numero di qubit nel sistema.
- `amplitudes`: Puntatore a un array di ampiezze complesse di dimensione $2^n$, dove $n$ é il numero di qubit.

Ogni stato quantistico di un sistema a $n$ qubit può essere descritto come una combinazione lineare degli stati base del sistema. Gli stati base di un sistema a $n$ qubit sono $2^n$, e sono rappresentati dalle combinazioni di tutti i possibili valori binari dei qubit, da $|00\ldots0\rangle$ a $|11\ldots1\rangle$.

Matematicamente, lo stato quantistico generale $|\psi\rangle$ può essere espresso come:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle$$

dove:

- $\alpha_i$ sono coefficienti complessi (ampiezze di probabilità) tali che $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$.
- $|i\rangle$ rappresenta lo stato base corrispondente al numero binario $i$.

Nel codice, l'array `amplitudes` contiene queste ampiezze $\alpha_i$, dove ogni elemento dell'array corrisponde a uno stato base specifico.

### Mappatura tra Stati Quantistici e Indici dell'Array

Per comprendere come gli stati quantistici sono mappati agli indici dell'array `amplitudes`, consideriamo che ogni stato base $|i\rangle$ é identificato dal suo indice $i$, che varia da 0 a $2^n - 1$.

Ad esempio, per un sistema a $n = 3$ qubit, abbiamo $2^3 = 8$ stati base:

| Indice $i$ | Stato base $\|i\rangle$ |
|---|---|
| 0 | $\|000\rangle$ |
| 1 | $\|001\rangle$ |
| 2 | $\|010\rangle$ |
| 3 | $\|011\rangle$ |
| 4 | $\|100\rangle$ |
| 5 | $\|101\rangle$ |
| 6 | $\|110\rangle$ |
| 7 | $\|111\rangle$ |

L'array `amplitudes` avrà quindi 8 elementi, dove `amplitudes[0]` corrisponde all'ampiezza $\alpha_0$ dello stato $|000\rangle$, `amplitudes[1]` corrisponde a $\alpha_1$ dello stato $|001\rangle$, e così via fino a `amplitudes[7]` per lo stato $|111\rangle$.

### Esempio di Inizializzazione dello Stato

Per inizializzare il sistema nello stato $|0\rangle^{\otimes n}$ (tutti i qubit in $|0\rangle$), impostiamo `amplitudes[0]` a 1.0 e tutti gli altri elementi a 0.

```
QubitState* state = initializeState(n);  // n \'e il numero di qubit
```

Listing A.2: Inizializzazione dello stato $|0\rangle^{\otimes n}$

La funzione `initializeState` alloca memoria per l'array `amplitudes` di dimensione $2^n$ e imposta `amplitudes[0]` a $1.0 + 0.0 * I$, mentre tutti gli altri elementi sono inizializzati a $0.0 + 0.0 * I$.

### Manipolazione degli Stati Quantistici

Quando applichiamo un gate quantistico, eseguiamo un'operazione unitaria che trasforma lo stato quantistico del sistema. Questa trasformazione comporta la moltiplicazione della matrice unitaria del gate con il vettore delle ampiezze attuali, calcolando così le nuove ampiezze di probabilità per ciascuno stato base. Nel codice di **QuantumSim**, le funzioni dedicate all'applicazione dei gate eseguono direttamente questa trasformazione aggiornando i valori nell'array `amplitudes`, che rappresenta le ampiezze complesse associate a ogni possibile stato del sistema quantistico.

**Esempio: Applicazione del Gate Hadamard su un Singolo Qubit** Consideriamo l'applicazione del gate Hadamard al qubit $k$ in un sistema a $n$ qubit. Il gate Hadamard ha la seguente matrice unitaria:

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

Quando applichiamo $H$ al qubit $k$, aggiorniamo le ampiezze degli stati in cui il qubit $k$ é $|0\rangle$ e $|1\rangle$ secondo la seguente regola:

$$\alpha'_i = \frac{1}{\sqrt{2}}(\alpha_i + \alpha_j)$$

$$\alpha'_j = \frac{1}{\sqrt{2}}(\alpha_i - \alpha_j)$$

dove $i$ e $j$ sono gli indici degli stati che differiscono solo nel valore del qubit $k$.

Nel codice, la funzione `applyHadamard` implementa questa trasformazione aggiornando direttamente le ampiezze nell'array `amplitudes`.

**Esempio Dettagliato: Applicazione del Gate Hadamard su un Singolo Qubit**

Consideriamo un sistema a $n = 1$ qubit inizializzato nello stato $|0\rangle$.

1. Lo stato iniziale é:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle = 1 \cdot |0\rangle + 0 \cdot |1\rangle$$

2. L'array `amplitudes` contiene:

   - `amplitudes[0]` $= 1.0 + 0.0 * I$ (corrisponde a $\alpha_0$)
   - `amplitudes[1]` $= 0.0 + 0.0 * I$ (corrisponde a $\alpha_1$)

3. Applichiamo il gate Hadamard al qubit 0:

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

4. Le nuove ampiezze sono:

$$\alpha'_0 = \frac{1}{\sqrt{2}}(\alpha_0 + \alpha_1) = \frac{1}{\sqrt{2}}(1 + 0) = \frac{1}{\sqrt{2}}$$

$$\alpha'_1 = \frac{1}{\sqrt{2}}(\alpha_0 - \alpha_1) = \frac{1}{\sqrt{2}}(1 - 0) = \frac{1}{\sqrt{2}}$$

5. L'array `amplitudes` viene aggiornato:

   - `amplitudes[0]` $= 0.7071 + 0.0 * I$
   - `amplitudes[1]` $= 0.7071 + 0.0 * I$

6. Lo stato quantistico dopo l'applicazione del gate Hadamard é:

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

### Scalabilità per Sistemi con Molti Qubit

Per sistemi con un numero maggiore di qubit, l'array `amplitudes` cresce esponenzialmente. Ad esempio, per $n = 20$ qubit, l'array avrà $2^{20} \approx 1$ milione di elementi.

Questo impone limiti pratici alla dimensione dei sistemi che possiamo simulare, poiché la memoria richiesta diventa significativa. É quindi importante gestire la memoria in modo efficiente e considerare ottimizzazioni, come la rappresentazione sparsa degli stati quantistici quando applicabile.

### Implementazione delle Operazioni di Misurazione

Quando misuriamo un qubit, l'array `amplitudes` viene modificato per riflettere il collasso dello stato quantistico. Le ampiezze degli stati incompatibili con il risultato della misurazione vengono azzerate, e le altre vengono rinormalizzate.

Ad esempio, se misuriamo il qubit $k$ e otteniamo il risultato 0, tutte le ampiezze degli stati in cui il qubit $k$ é 1 vengono azzerate.

### Esempio di Misurazione

Continuando l'esempio precedente, supponiamo di misurare il qubit 0 dopo aver applicato il gate Hadamard.

- Le probabilità di ottenere 0 o 1 sono entrambe $|\alpha'_0|^2 = |\alpha'_1|^2 = \left(\frac{1}{\sqrt{2}}\right)^2 = 0.5$.
- Supponiamo che la misura dia risultato 0.
- Aggiorniamo le ampiezze:
  - `amplitudes[0]` viene rinormalizzata: $\alpha''_0 = \alpha'_0/\sqrt{0.5} = 1.0$
  - `amplitudes[1]` viene azzerata: $\alpha''_1 = 0.0$
- Lo stato quantistico dopo la misura é tornato a $|0\rangle$.

### Vantaggi della Rappresentazione Utilizzata

La rappresentazione dello stato quantistico tramite l'array `amplitudes` offre diversi vantaggi:

- **Semplicità**: É una rappresentazione diretta e intuitiva degli stati quantistici.
- **Flessibilità**: Permette di implementare qualsiasi operazione unitaria aggiornando le ampiezze.

- **Accesso Diretto**: Facilita l'accesso e la manipolazione delle ampiezze per specifici stati base.

Tuttavia, come menzionato, il principale svantaggio é la crescita esponenziale della memoria richiesta con l'aumentare del numero di qubit.

**Soluzioni Alternative per Ridurre l'Uso di Memoria** Per affrontare questa limitazione, esistono diverse strategie che permettono di ridurre l'uso di memoria necessaria per la simulazione di sistemi quantistici di grandi dimensioni:

- **Rappresentazioni Sparse**: Quando lo stato quantistico é sparso, ovvero quando molte delle ampiezze sono zero o trascurabili, si può utilizzare una rappresentazione sparsa che memorizza solo le ampiezze non nulle. Questo approccio riduce significativamente la quantità di memoria necessaria.
  - **Svantaggi**: Non é efficace per stati altamente entangled o distribuiti, dove molte ampiezze sono non nulle. Inoltre, la gestione di strutture dati sparse può aumentare la complessità computazionale delle operazioni sui gate.
- **Reti di Tensori**: Tecniche come le reti di tensori, inclusi i **Matrix Product States** (MPS) o le **Projected Entangled Pair States** (PEPS), permettono di rappresentare stati quantistici in modo più efficiente sfruttando la struttura dell'entanglement. Queste rappresentazioni possono ridurre la complessità della memoria da esponenziale a polinomiale in alcuni casi.
  - **Svantaggi**: Le reti di tensori possono essere complesse da implementare e gestire. Inoltre, non sono adatte per tutti i tipi di stati quantistici, specialmente quelli con un elevato grado di entanglement.
- **Simulazioni Basate su Circuiti**: Anziché memorizzare l'intero stato quantistico, alcune simulazioni si basano sull'applicazione sequenziale dei gate quantistici senza mantenere esplicitamente tutte le ampiezze. Questo approccio può ridurre l'uso di memoria, ma potrebbe richiedere più tempo di calcolo.
  - **Svantaggi**: Potrebbe essere meno efficiente in termini di tempo di esecuzione, specialmente per circuiti complessi. Inoltre, la mancanza di accesso diretto allo stato quantistico può limitare alcune operazioni di analisi e debug.
- **Parallelizzazione e Distribuzione della Memoria**: Utilizzare tecniche di parallel computing e distribuire la memoria su più nodi di calcolo permette di gestire sistemi con un numero maggiore di qubit, sfruttando risorse computazionali distribuite.
  - **Svantaggi**: La parallelizzazione richiede una programmazione più complessa e può introdurre overhead di comunicazione tra i nodi. Inoltre, non tutti i problemi quantistici si prestano bene alla parallelizzazione.

- **Compressione dello Stato Quantistico**: Algoritmi di compressione possono essere applicati per ridurre la dimensione dell'array delle ampiezze, mantenendo una rappresentazione approssimata dello stato quantistico.
  - **Svantaggi**: La compressione introduce approssimazioni che possono compromettere l'accuratezza delle simulazioni. Inoltre, la decompressione e la gestione delle rappresentazioni compresse possono aggiungere complessità computazionale.

Queste soluzioni offrono vie promettenti per superare le limitazioni imposte dalla crescita esponenziale della memoria, permettendo di simulare sistemi quantistici più complessi e realistici. Tuttavia, ciascuna di esse comporta compromessi in termini di accuratezza, efficienza computazionale o generalità, e la scelta della tecnica appropriata dipende dalle specifiche esigenze della simulazione.

**Esempi di Simulatori che Utilizzano Approcci Alternativi** Esistono diversi simulatori quantistici che adottano le soluzioni alternative descritte per ridurre l'uso di memoria. Di seguito vengono presentati alcuni esempi significativi:

- **QuTiP (Quantum Toolbox in Python)**:
  - **Approccio**: Rappresentazioni Sparse e Matrici Densità.
  - **Descrizione**: QuTiP é una libreria open-source in Python progettata per la simulazione di sistemi quantistici. Supporta rappresentazioni sparse degli stati quantistici, consentendo di simulare sistemi con un gran numero di qubit quando gli stati sono sparsi. Inoltre, QuTiP permette la simulazione di stati misti utilizzando matrici densità, utile per modellare la decoerenza e il rumore.
- **qHipster**:
  - **Approccio**: Reti di Tensori (Tensor Networks).
  - **Descrizione**: qHipster é un simulatore quantistico basato su reti di tensori, sviluppato per sfruttare la struttura dell'entanglement negli stati quantistici. Utilizza tecniche avanzate di decomposizione tensoriale per rappresentare stati altamente entangled in modo efficiente, riducendo significativamente l'uso di memoria rispetto alla rappresentazione completa dello stato.
- **ProjectQ**:
  - **Approccio**: Simulazioni Basate su Circuiti.
  - **Descrizione**: ProjectQ é un framework open-source per il calcolo quantistico che supporta diverse backends di simulazione, inclusi quelli basati su circuiti quantistici. Questo approccio permette di eseguire simulazioni senza dover memorizzare l'intero stato quantistico, applicando sequenzialmente i gate e calcolando i risultati in tempo reale.

- **qHipster**:
  - **Approccio**: Reti di Tensori.
  - **Descrizione**: qHipster é un simulatore quantistico che utilizza le reti di tensori per rappresentare e manipolare stati quantistici. Questo metodo sfrutta la struttura dell'entanglement per comprimere le informazioni dello stato, permettendo la simulazione di sistemi con un numero maggiore di qubit rispetto alle rappresentazioni complete.
- **LIQUi|>**:
  - **Approccio**: Simulazioni Basate su Circuiti e Parallelizzazione.
  - **Descrizione**: LIQUi|> é un ambiente di simulazione quantistica open-source sviluppato da Microsoft. Supporta diversi backend di simulazione, inclusi quelli che sfruttano la parallelizzazione e la distribuzione della memoria per gestire sistemi con un elevato numero di qubit. Questo permette di scalare le simulazioni utilizzando risorse computazionali distribuite.
- **TensorCircuit**:
  - **Approccio**: Reti di Tensori.
  - **Descrizione**: TensorCircuit é un simulatore quantistico che utilizza reti di tensori per rappresentare stati quantistici. Implementa tecniche di decomposizione tensoriale per gestire l'entanglement in modo efficiente, riducendo l'uso di memoria e migliorando le prestazioni nelle simulazioni di circuiti complessi.
- **Yao.jl**:
  - **Approccio**: Rappresentazioni Sparse e Parallelizzazione.
  - **Descrizione**: Yao.jl é un framework di simulazione quantistica sviluppato in Julia. Supporta rappresentazioni sparse degli stati quantistici e sfrutta la parallelizzazione per migliorare le prestazioni nelle simulazioni. Questo consente di simulare circuiti con un numero elevato di qubit, specialmente quando lo stato é sparso.

**Considerazioni sugli Svantaggi delle Rappresentazioni Alternative** Nonostante le rappresentazioni alternative offrano vantaggi significativi nella riduzione dell'uso di memoria, presentano anche alcuni svantaggi:

- **Rappresentazioni Sparse**:
  - **Svantaggi**: Non sono efficaci per stati altamente entangled o distribuiti, dove molte ampiezze sono non nulle. Inoltre, la gestione di strutture dati sparse può aumentare la complessità computazionale delle operazioni sui gate.

- **Reti di Tensori**:
  - **Svantaggi**: Le reti di tensori possono essere complesse da implementare e gestire. Non sono adatte per tutti i tipi di stati quantistici, specialmente quelli con un elevato grado di entanglement.
- **Simulazioni Basate su Circuiti**:
  - **Svantaggi**: Potrebbe essere meno efficiente in termini di tempo di esecuzione, specialmente per circuiti complessi. Inoltre, la mancanza di accesso diretto allo stato quantistico può limitare alcune operazioni di analisi e debug.
- **Parallelizzazione e Distribuzione della Memoria**:
  - **Svantaggi**: La parallelizzazione richiede una programmazione più complessa e può introdurre overhead di comunicazione tra i nodi. Inoltre, non tutti i problemi quantistici si prestano bene alla parallelizzazione.
- **Compressione dello Stato Quantistico**:
  - **Svantaggi**: La compressione introduce approssimazioni che possono compromettere l'accuratezza delle simulazioni. Inoltre, la decompressione e la gestione delle rappresentazioni compresse possono aggiungere complessità computazionale.

Questi svantaggi evidenziano la necessità di scegliere l'approccio più appropriato in base alle specifiche esigenze della simulazione e al tipo di circuito quantistico da analizzare. La scelta dipende da vari fattori, tra cui il numero di qubit, il grado di entanglement, e la precisione richiesta nelle simulazioni.

#### Vantaggi della Rappresentazione Utilizzata

La rappresentazione dello stato quantistico tramite l'array `amplitudes` offre diversi vantaggi:

- **Semplicità**: É una rappresentazione diretta e intuitiva degli stati quantistici.
- **Flessibilità**: Permette di implementare qualsiasi operazione unitaria aggiornando le ampiezze.
- **Accesso Diretto**: Facilita l'accesso e la manipolazione delle ampiezze per specifici stati base.

Tuttavia, come menzionato, il principale svantaggio é la crescita esponenziale della memoria richiesta con l'aumentare del numero di qubit.

### Considerazioni sull'Implementazione in C

Essendo il linguaggio C un linguaggio a basso livello, la gestione della memoria e dei puntatori richiede attenzione. Nel codice di **QuantumSim**:

- La memoria per `amplitudes` viene allocata dinamicamente usando `malloc` o `calloc`.
- É fondamentale liberare la memoria con `free` per evitare perdite di memoria (*memory leaks*).
- Si utilizzano tipi di dati appropriati (`double complex`) per gestire i numeri complessi, sfruttando le librerie standard del C come `<complex.h>`.

### Esempio Completo: Inizializzazione, Applicazione di Gate e Misurazione

Mettiamo insieme i concetti con un esempio completo in codice:

```c
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
    // Inizializza un sistema con 2 qubit
    QubitState* state = initializeState(2);

    // Applica Hadamard al qubit 0
    applyHadamard(state, 0);

    // Applica CNOT con controllo su qubit 0 e target su qubit 1
    applyCNOT(state, 0, 1);

    // Stampa lo stato quantistico
    printState(state);

    // Misura il qubit 0
    MeasurementResult result0 = measure(state, 0);

    // Misura il qubit 1
    MeasurementResult result1 = measure(state, 1);

    printf("Risultati delle misure: qubit 0 = %d, qubit 1 = %d\n", result0.
    result, result1.result);

    // Libera la memoria
    freeState(state);

    return 0;
}
```

Listing A.3: circuit10.1.c

Questo programma:

1. Inizializza lo stato $|00\rangle$.
2. Crea uno stato entangled applicando Hadamard e CNOT, ottenendo lo stato di Bell $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.
3. Stampa le ampiezze degli stati.
4. Misura entrambi i qubit, che daranno risultati correlati a causa dell'entanglement.

#### Interpretazione dei Risultati

Poiché i qubit sono entangled, le misure daranno risultati correlati:

- Se il qubit 0 é misurato in $|0\rangle$, anche il qubit 1 sarà $|0\rangle$.
- Se il qubit 0 é misurato in $|1\rangle$, anche il qubit 1 sarà $|1\rangle$.

Questo esempio illustra come la struttura `QubitState` e l'array `amplitudes` consentano di rappresentare e manipolare stati quantistici complessi, inclusi gli stati entangled.

### A.3.3 Applicazione dei Gate Quantistici

I gate quantistici sono implementati attraverso funzioni che modificano le ampiezze dello stato quantistico.

#### Gate a Singolo Qubit

La funzione generica per applicare un gate a singolo qubit é:

```
void applySingleQubitGate(QubitState *state, int target, double complex
    gate[2][2]);
```

#### Esempio: Applicazione del Gate Hadamard

```
void applyHadamard(QubitState *state, int target) {
    double complex H[2][2] = {
        {1.0 / sqrt(2.0),  1.0 / sqrt(2.0)},
        {1.0 / sqrt(2.0), -1.0 / sqrt(2.0)}
    };
    applySingleQubitGate(state, target, H);
}
```

Listing A.4: Esempio di utilizzo di QuantumSim

#### Meccanismo di Funzionamento

- Per ogni stato base, determina se il qubit target é in $|0\rangle$ o $|1\rangle$.
- Calcola le nuove ampiezze applicando la matrice del gate.
- Aggiorna le ampiezze dello stato quantistico.

#### Gate a Due e Tre Qubit

I gate a più qubit, come CNOT e Toffoli, utilizzano funzioni specifiche che gestiscono le interazioni tra i qubit di controllo e il qubit target.

#### Esempio: Applicazione del Gate CNOT

```
void applyCNOT(QubitState *state, int control, int target) {
    // Implementazione del gate CNOT
}
```

Listing A.5: Esempio di utilizzo di QuantumSim

#### Meccanismo di Funzionamento

- Scorre tutti gli stati base possibili.
- Identifica gli stati in cui il qubit di controllo é $|1\rangle$.
- Inverte il qubit target in quegli stati.

### A.3.4 Misurazione dei Qubit

La misurazione é implementata in modo da simulare il collasso dello stato quantistico.

```
MeasurementResult measure(QubitState *state, int qubit);
```

Listing A.6: Esempio di utilizzo di QuantumSim

#### Processo di Misurazione

- Calcola la probabilità di ottenere 0 o 1 per il qubit misurato.
- Genera un numero casuale per determinare il risultato della misura.
- Collassa lo stato quantistico in base al risultato, eliminando le componenti non coerenti con la misura.

### A.3.5 Gestione della Memoria e Pulizia

La funzione `freeState` é utilizzata per liberare la memoria allocata per lo stato quantistico:

```
void freeState(QubitState *state) {
    free(state->amplitudes);
    free(state);
}
```

Listing A.7: Esempio di utilizzo di QuantumSim

**Importanza** La gestione corretta della memoria é cruciale in C per evitare perdite di memoria (*memory leaks*) che possono compromettere le prestazioni del simulatore.

### A.3.6 Interazione tra le Componenti

L'architettura del **QuantumSim** é progettata per essere modulare e estensibile. Le diverse funzioni interagiscono tra loro in modo coerente:

- Le funzioni di inizializzazione preparano lo stato quantistico.
- Le funzioni dei gate modificano lo stato in base alle operazioni desiderate.
- Le funzioni di misurazione permettono di estrarre informazioni dal sistema.
- La gestione della memoria assicura che le risorse siano utilizzate in modo efficiente.

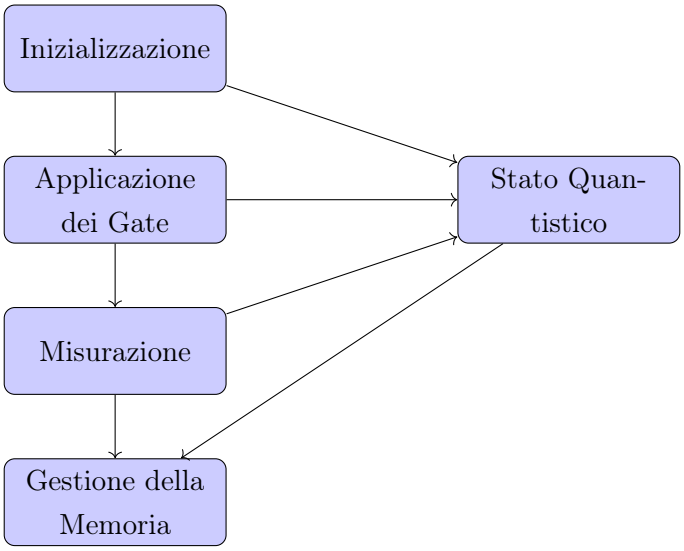
### A.3.7 Estensibilità del Simulatore

Grazie alla sua architettura modulare, il **QuantumSim** può essere esteso facilmente:

- **Aggiunta di Nuovi Gate**: É possibile implementare nuovi gate definendo la matrice unitaria corrispondente e utilizzando le funzioni esistenti per applicarla.
- **Simulazione di Rumore**: Si possono introdurre funzioni che simulano la decoerenza o altri effetti ambientali.
- **Ottimizzazione delle Prestazioni**: Utilizzando tecniche avanzate, come la parallelizzazione, per migliorare l'efficienza.

### A.3.8 Diagramma dell'Architettura

Per visualizzare l'architettura del **QuantumSim**, possiamo rappresentare le componenti principali e le loro interazioni con un diagramma.



**Legenda**

- **Inizializzazione**: Preparazione dello stato iniziale.
- **Applicazione dei Gate**: Modifica dello stato attraverso operazioni quantistiche.
- **Misurazione**: Estrazione di informazioni dal sistema.
- **Gestione della Memoria**: Allocazione e liberazione delle risorse.
- **Stato Quantistico**: Rappresenta il sistema quantistico simulato.

## A.4 Inizializzazione e Manipolazione di Qubit Singoli

In questa sezione, esploreremo come inizializzare lo stato quantistico e applicare gate a singolo qubit utilizzando il nostro simulatore.

### A.4.1 Inizializzare lo Stato Quantistico

Per inizializzare un sistema quantistico con $n$ qubit nello stato $|0\rangle^{\otimes n}$, utilizziamo la funzione:

```
QubitState* state = initializeState(numQubits);
```

Listing A.8: Esempio di utilizzo di QuantumSim

Dove `numQubits` é il numero di qubit nel sistema.

Per impostare un qubit specifico nello stato $|1\rangle$, possiamo usare:

```
initializeSingleQubitToOne(state, targetQubit);
```

Listing A.9: Esempio di utilizzo di QuantumSim

### A.4.2 Applicazione di Gate a Singolo Qubit

I gate a singolo qubit sono fondamentali per manipolare lo stato dei qubit individualmente.

#### Gate Hadamard

Il gate Hadamard trasforma lo stato $|0\rangle$ in una sovrapposizione equa di $|0\rangle$ e $|1\rangle$:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Per applicare il gate Hadamard a un qubit nel simulatore:

```
applyHadamard(state, targetQubit);
```

Listing A.10: Esempio di utilizzo di QuantumSim

**Rappresentazione Matriciale** La matrice del gate Hadamard é:

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

### Esempio Pratico

Creiamo un semplice circuito che prepara un qubit in sovrapposizione e poi lo misura.

### Codice

```c
#include "quantum_sim.h"
#include <stdio.h>

int circuit(void) {
    QubitState* state = initializeState(1);
    applyHadamard(state, 0);
    MeasurementResult result = measure(state, 0);
    printf("Risultato della misura: %d\n", result.result);
    freeState(state);
    return 0;
}
```

Listing A.11: circuit10.2.c

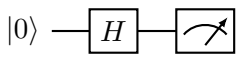
### Spiegazione

- Inizializziamo un sistema con 1 qubit nello stato $|0\rangle$.
- Applichiamo il gate Hadamard al qubit, creando una sovrapposizione:

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

- Misuriamo il qubit, che avrà una probabilità del 50% di essere $|0\rangle$ e del 50% di essere $|1\rangle$.

### Diagramma del Circuito




## A.5 Operazioni su Qubit Multipli

Esploreremo ora come manipolare sistemi con più qubit, introducendo gate a due e tre qubit e creando stati entangled.

### A.5.1 Gate a Due Qubit: CNOT

Il gate CNOT (Controlled-NOT) é un gate fondamentale che agisce su due qubit: un qubit di controllo e un qubit target.

**Funzionamento** Il gate CNOT inverte il qubit target se e solo se il qubit di controllo é nello stato $|1\rangle$.

#### Applicazione nel Simulatore

```
applyCNOT(state, controlQubit, targetQubit);
```

Listing A.12: Esempio di utilizzo di QuantumSim

#### Rappresentazione Matriciale

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

### A.5.2 Creazione di Stati Entangled

Gli stati entangled sono stati quantistici in cui i qubit non possono essere descritti indipendentemente.

#### Esempio Pratico: Stato di Bell

Creiamo uno stato di Bell utilizzando i gate Hadamard e CNOT.

#### Codice

```
QubitState* state = initializeState(2);
applyHadamard(state, 0);
applyCNOT(state, 0, 1);
printState(state);
```

Listing A.13: Esempio di utilizzo di QuantumSim

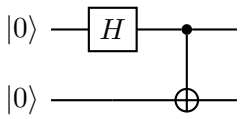
#### Spiegazione

- Inizializziamo un sistema con 2 qubit nello stato $|00\rangle$.
- Applichiamo il gate Hadamard al qubit 0:

$$|0\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

- Applichiamo il gate CNOT con qubit 0 come controllo e qubit 1 come target.
- Lo stato finale é:
$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$
- Questo é uno dei quattro stati di Bell, che rappresentano massima entanglement tra due qubit.

**Diagramma del Circuito**




**Visualizzazione dello Stato**

La funzione `printState` mostrerà le ampiezze degli stati base:

```
Stato 0: 0.707107 + 0.000000i | 00
Stato 1: 0.000000 + 0.000000i | 01
Stato 2: 0.000000 + 0.000000i | 10
Stato 3: 0.707107 + 0.000000i | 11
```

Listing A.14: Esempio di utilizzo di QuantumSim

## A.6 Implementazione di Algoritmi Quantistici Semplici

Esploreremo ora l'implementazione di algoritmi quantistici fondamentali utilizzando il nostro simulatore. Alcuni di questi algoritmi (come il seguente) sono stati già proposti come esercizi nel testo. Qui li vedremo più dal punto di vista informatico che non da quello fisico, focalizzando l'attenzione su come questi sono implementati usando questo specifico simulatore. Nel caso specifico, per esempio, implementeremo solo la funzione bilanciata per poi analizzare il codice.

### A.6.1 Algoritmo di Deutsch-Jozsa

**Descrizione Teorica**

L'algoritmo di Deutsch-Jozsa determina se una funzione booleana $f : \{0,1\}^n \to \{0,1\}$ é **costante** (restituisce lo stesso valore per tutte le possibili input) o **bilanciata** (restituisce 0 per metà delle input e 1 per l'altra metà) con un solo calcolo quantistico.

**Implementazione Pratica**

```
#include "quantum_sim.h"
#include <stdio.h>

```

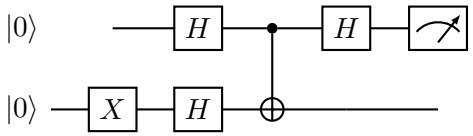
```c
int circuit(void) {
    QubitState* state = initializeState(2);
    applyX(state,1);
    // Applica Hadamard ai qubit
    applyHadamard(state, 0); //|+>
    applyHadamard(state, 1); //|->
    // Oracolo per una funzione bilanciata
    applyCNOT(state, 0, 1);
    // Applica Hadamard al qubit di input
    applyHadamard(state, 0);
    MeasurementResult result = measure(state, 0);
    if (result.result == 0) {
        printf("La funzione \'e costante.\n");
    } else {
        printf("La funzione \'e bilanciata.\n");
    }
    freeState(state);
}
```

Listing A.15: circuit10.3.c

### Spiegazione

- Inizializziamo un sistema con 2 qubit: il primo é il qubit di input a 0, il secondo é un qubit ausiliario a 1.
- Applichiamo il gate Hadamard a entrambi i qubit per creare una sovrapposizione di tutti gli stati possibili.
- Implementiamo l'oracolo che rappresenta la funzione $f$. In questo caso, usiamo un CNOT per una funzione bilanciata.
- Applichiamo nuovamente il gate Hadamard al qubit di input.
- Misuriamo il qubit di input. Se il risultato é $|0\rangle$, la funzione é costante; altrimenti, é bilanciata.

### Diagramma del Circuito




### Analisi dei Risultati

Questo algoritmo mostra un vantaggio quantistico poiché determina con certezza se la funzione é costante o bilanciata con un solo calcolo, mentre un algoritmo classico richiederebbe un numero di valutazioni esponenziale in $n$.

## A.7 Costruzione di Circuiti Complessi

Esploreremo ora gate più complessi e come possono essere utilizzati per costruire circuiti avanzati.

### A.7.1 Gate Toffoli

Il **gate Toffoli** é un gate a tre qubit con due qubit di controllo e un qubit target. Inverte il qubit target se e solo se entrambi i qubit di controllo sono nello stato $|1\rangle$.

#### Applicazione nel Simulatore

```
applyToffoli(state, controlQubit1, controlQubit2, targetQubit);
```

Listing A.16: Esempio di utilizzo di QuantumSim

#### Esempio Pratico: Funzione AND Logica

Implementiamo una funzione AND utilizzando il gate Toffoli.

#### Codice

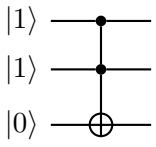
```
QubitState* state = initializeState(3);
initializeSingleQubitToOne(state, 0); // Primo controllo
initializeSingleQubitToOne(state, 1); // Secondo controllo
applyToffoli(state, 0, 1, 2);
printState(state);
```

Listing A.17: Esempio di utilizzo di QuantumSim

#### Spiegazione

- Impostiamo i qubit di controllo (0 e 1) nello stato $|1\rangle$.
- Il qubit target (2) é inizialmente in $|0\rangle$.
- Applichiamo il gate Toffoli. Il qubit target sarà invertito a $|1\rangle$ poiché entrambi i controlli sono $|1\rangle$.

#### Diagramma del Circuito



### A.7.2 Gate di Fase Controllati

I gate di fase controllati introducono una fase allo stato del qubit target condizionato dallo stato dei qubit di controllo.

#### Gate CCZ

Il gate CCZ applica un gate Z al target se entrambi i qubit di controllo sono $|1\rangle$.

#### Applicazione nel Simulatore

```
applyCCZ(state, controlQubit1, controlQubit2, targetQubit);
```

Listing A.18: Esempio di utilizzo di QuantumSim

### A.7.3 Implementazione di Funzioni Logiche Classiche

Utilizzando i gate quantistici, possiamo implementare funzioni logiche classiche in modo reversibile, essenziale nel calcolo quantistico.

## A.8 Misurazione e Interpretazione dei Risultati

La misurazione é un aspetto cruciale del calcolo quantistico, poiché é l'atto di misurare che ci permette di estrarre informazioni dallo stato quantistico.

### A.8.1 Processo di Misurazione

La misurazione di un qubit collassa lo stato quantistico in uno degli autostati dell'operatore di misurazione. La probabilità di ottenere un particolare risultato é data dal quadrato del modulo dell'ampiezza associata a quello stato.

### A.8.2 Funzioni di Misurazione nel Codice

Per misurare un singolo qubit:

```
MeasurementResult result = measure(state, qubitIndex);
```

Listing A.19: Esempio di utilizzo di QuantumSim

Il `MeasurementResult` contiene:

- `result`: il risultato della misura (0 o 1).
- `prob0`: la probabilità di ottenere 0.
- `prob1`: la probabilità di ottenere 1.

Per misurare tutti i qubit:

```
int* results = measure_all(state);
```

Listing A.20: Esempio di utilizzo di QuantumSim

### A.8.3 Esempio Pratico

#### Codice

```
QubitState* state = initializeState(2);
applyHadamard(state, 0);
applyHadamard(state, 1);
int* results = measure_all(state);
printf("Risultati delle misure: %d %d\n", results[0], results[1]);
free(results);
freeState(state);
```

Listing A.21: Esempio di utilizzo di QuantumSim

#### Spiegazione

- Creiamo una sovrapposizione di tutti gli stati possibili applicando Hadamard a entrambi i qubit.
- Misuriamo entrambi i qubit. Ciascuno ha una probabilità del 50% di essere 0 o 1.
- Stampiamo i risultati delle misure.

## A.9 Esercizi Pratici

Per consolidare la comprensione, proponiamo un esercizio pratico di implementazione.

### A.9.1 Implementare il Teletrasporto Quantistico

#### Descrizione dell'Algoritmo

Il teletrasporto quantistico permette di trasferire lo stato di un qubit da una posizione a un'altra utilizzando entanglement e comunicazione classica.

#### Guida all'Implementazione

1. **Preparazione degli Stati**

   Inizializziamo un sistema con 3 qubit:

   - Qubit 0: il qubit da teletrasportare, inizializzato in uno stato arbitrario.
   - Qubit 1 e 2: coppia entangled condivisa tra mittente e destinatario.

2. **Creazione dell'Entanglement**

   Applichiamo Hadamard e CNOT per creare l'entanglement tra qubit 1 e 2:

```
    applyHadamard(state, 1);
    applyCNOT(state, 1, 2);

```

Listing A.22: Esempio di utilizzo di QuantumSim

3. **Applicazione delle Operazioni di Bell**

   Eseguiamo operazioni sul qubit da teletrasportare e sul qubit del mittente:

```
    applyCNOT(state, 0, 1);
    applyHadamard(state, 0);

```

Listing A.23: Esempio di utilizzo di QuantumSim

4. **Misurazione dei Qubit 0 e 1**

   Misuriamo i qubit 0 e 1 e salviamo i risultati:

```
    int m0 = measure(state, 0).result;
    int m1 = measure(state, 1).result;

```

Listing A.24: Esempio di utilizzo di QuantumSim

5. **Operazioni Correttive sul Qubit 2**

   In base ai risultati delle misure, applichiamo operazioni correttive al qubit 2:

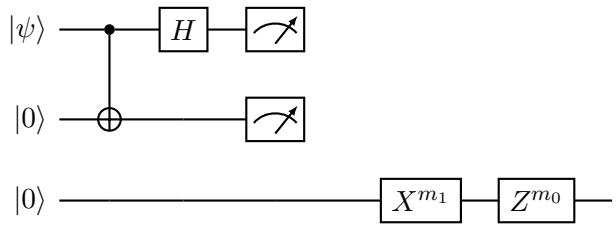
```
    if (m1 == 1) {
        applyX(state, 2);
    }
    if (m0 == 1) {
        applyZ(state, 2);
    }

```

Listing A.25: Esempio di utilizzo di QuantumSim

6. **Verifica del Teletrasporto**

   Lo stato iniziale del qubit 0 é stato trasferito al qubit 2. Possiamo confrontare gli stati per verificare il successo.

**Diagramma del Circuito**




Dove $m_0$ e $m_1$ sono i risultati delle misure dei qubit 0 e 1.

**Spiegazione Dettagliata**

- **Fase 1**: Il qubit 0 é nello stato $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.
- **Fase 2**: I qubit 1 e 2 sono preparati in uno stato di Bell $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$.
- **Fase 3**: Si eseguono operazioni che entanglano il qubit 0 con i qubit 1 e 2.
- **Fase 4**: Misurando i qubit 0 e 1, lo stato del qubit 2 diventa una versione trasformata di $|\psi\rangle$.
- **Fase 5**: Le operazioni correttive riportano il qubit 2 allo stato originale $|\psi\rangle$.

## A.10 Approfondimento sul Codice del Simulatore

### A.10.1 Struttura dei Dati

Lo stato quantistico é rappresentato dalla struttura `QubitState`:

- `numQubits`: Numero di qubit nel sistema.
- `amplitudes`: Array di ampiezze complesse di dimensione $2^n$, dove $n$ é il numero di qubit.

### A.10.2 Implementazione dei Gate

I gate quantistici sono implementati come operazioni matriciali che modificano le ampiezze dello stato quantistico.

**Esempio: Gate Hadamard**

```
void applyHadamard(QubitState *state, int target) {
    double complex H[2][2] = {
        {1.0 / sqrt(2.0),  1.0 / sqrt(2.0)},
        {1.0 / sqrt(2.0), -1.0 / sqrt(2.0)}
    };
    applySingleQubitGate(state, target, H);
}
```

Listing A.26: Esempio di utilizzo di QuantumSim

**Spiegazione** La funzione `applySingleQubitGate` applica il gate specificato al qubit target, aggiornando le ampiezze dello stato quantistico in base alla matrice del gate.

### A.10.3 Ottimizzazioni Possibili

- **Parallelizzazione**: Utilizzare tecniche di parallel computing per gestire sistemi con un numero elevato di qubit.
- **Rappresentazione Sparsa**: Per stati quantistici sparsi, utilizzare strutture dati che memorizzano solo le ampiezze non nulle, riducendo l'utilizzo di memoria.
- **Ottimizzazione degli Algoritmi**: Implementare algoritmi più efficienti per l'applicazione dei gate e la gestione delle misurazioni.

### A.10.4 Possibili Miglioramenti e Progetti Futuri

Nonostante QuantumSim offra una solida base per la simulazione di circuiti quantistici, vi sono diverse aree in cui può essere ampliato e perfezionato:

- **Ottimizzazione delle Prestazioni**: Implementare algoritmi più efficienti e sfruttare il parallelismo hardware per gestire un numero maggiore di qubit.
- **Supporto per Stati Misti**: Estendere il simulatore per includere la simulazione di stati misti e canali quantistici, utilizzando matrici densità.
- **Simulazione di Decoerenza e Rumore**: Integrare modelli che rappresentino l'interazione con l'ambiente, rendendo le simulazioni più realistiche.
- **Interfaccia Utente Avanzata**: Sviluppare un'interfaccia grafica o migliorare l'API esistente per rendere il simulatore più accessibile e intuitivo.
- **Integrazione con Altri Strumenti**: Collegare QuantumSim ad altri software open source per ampliare le sue funzionalità e favorire la sperimentazione.

### A.10.5 Collaborazione Open Source

QuantumSim é un progetto libero e open source, rilasciato sotto la licenza GNU GPL. Questo significa che chiunque é libero di utilizzare, studiare, modificare e distribuire il software.

Invito la comunità a contribuire allo sviluppo di QuantumSim. La collaborazione può accelerare il progresso e portare a miglioramenti che beneficiano tutti. Che siate sviluppatori, ricercatori o semplici appassionati, il vostro contributo é prezioso.

Per partecipare al progetto, potete visitare il repository ufficiale all'indirizzo:

https://github.com/francescosisini/QuantumSim/

### A.10.6 Inclusione dei file sorgenti

Per rendere questo libro un'opera completa e indipendente, abbiamo incluso i file sorgenti direttamente all'interno del testo. I file rappresentano il codice utilizzato per sviluppare e dimostrare i concetti presentati nei capitoli precedenti. Tuttavia, è importante sottolineare che la versione più aggiornata del progetto, con eventuali miglioramenti o correzioni, è disponibile sul repository ufficiale GitHub.

Potete trovare il progetto completo su GitHub al seguente indirizzo:

`https://github.com/francescosisini/QuantumSim`

Si consiglia di scaricare i file direttamente dal repository per garantire di lavorare con l'ultima versione del codice. L'inclusione dei file sorgenti nel libro serve principalmente a fornire un riferimento immediato e a rendere l'opera autonoma per la lettura e lo studio offline.
Gli esempi inclusi coprono:

- Il simulatore QuantumSim e le sue funzioni principali.
- Codici di esempio per algoritmi quantistici, come l'algoritmo di Grover.
- Esempi pratici per esercizi di calcolo quantistico.

Invitiamo i lettori interessati a contribuire al progetto o a segnalare eventuali problemi tramite il repository GitHub. Questa collaborazione è fondamentale per mantenere il progetto attuale e rilevante.

```c
/*
 * QuantumSim: A Quantum Circuit Simulator for C Programmers
 * Copyright (C) 2024 Francesco Sisini
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <https://www.gnu.org/licenses/>.
 */

#ifndef QUANTUM_SIM_H
#define QUANTUM_SIM_H

#include <complex.h>
#include <math.h>

typedef struct {
    int numQubits;
    double complex *amplitudes;
} QubitState;

typedef struct {
    double prob0;  // Probabilit  di misurare 0
    double prob1;  // Probabilit  di misurare 1
    int result;    // Risultato della misurazione (0 o 1)
} MeasurementResult;


typedef struct {
    double complex amplitude0;
    double complex amplitude1;
} QubitAmplitudes;

QubitState* initializeState(int numQubits);
void initializeStateTo(QubitState *state, int index);
void initializeSingleQubitToOne(QubitState *state, int targetQubit);
void freeState(QubitState *state);
void printState(QubitState *state);
void printStateIgnoringQubits(QubitState *state, int *ignoreQubits, int
    numIgnoreQubits);
void applyHadamard(QubitState *state, int target);
void applyX(QubitState *state, int target);
```

```
50 void applyY(QubitState *state, int target);
51 void applyZ(QubitState *state, int target);
52 void applyT(QubitState *state, int target);
53 void applyTdag(QubitState *state, int target);
54 void applyS(QubitState *state, int target);
55 void applyCNOT(QubitState *state, int control, int target);
56 void applyCZ(QubitState *state, int control, int target);
57 void applyCPhaseShift(QubitState *state, int control, int target, double
       complex phase);
58 void applyPhase(QubitState* state, int qubit, double phase);
59 void applySingleQubitGate(QubitState *state, int target, double complex
       gate[2][2]);
60
61 // Nuove funzioni a 3-qubit
62 void applyFredkin(QubitState* state, int control, int target1, int target2)
       ;
63 void applyCCZ(QubitState* state, int control1, int control2, int target);
64 void applyToffoli(QubitState* state, int control1, int control2, int target
       );
65 void applyCCY(QubitState* state, int control1, int control2, int target);
66 void applyCCPhase(QubitState* state, int control1, int control2, int target
       , double phase);
67
68 // Misure e controlli
69 int* measure_all(QubitState *state);
70 MeasurementResult measure(QubitState *state, int qubit);
71 QubitAmplitudes getQubitAmplitudes(QubitState* state, int target);
72 void printQubitAmplitudes(QubitAmplitudes amplitudes);
73
74 #endif // QUANTUM_SIM_H
```

Listing A.27: Header

```
1  /*
2   * QuantumSim: A Quantum Circuit Simulator for C Programmers
3   * Copyright (C) 2024 Francesco Sisini
4   *
5   * This program is free software: you can redistribute it and/or modify
6   * it under the terms of the GNU General Public License as published by
7   * the Free Software Foundation, either version 3 of the License, or
8   * (at your option) any later version.
9   *
10  * This program is distributed in the hope that it will be useful,
11  * but WITHOUT ANY WARRANTY; without even the implied warranty of
12  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13  * GNU General Public License for more details.
14  *
15  * You should have received a copy of the GNU General Public License
16  * along with this program.  If not, see <https://www.gnu.org/licenses/>.
17  */
18
```

```c
#include "quantum_sim.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <complex.h>
#include <time.h>

#ifndef M_PI
    #define M_PI 3.14159265358979323846
#endif

/**
 * Inizializza lo stato quantistico a uno stato di base specifico.
 */
void initializeStateTo(QubitState *state, int index) {
    long long dim = 1LL << state->numQubits;
    for (long long i = 0; i < dim; i++) {
        state->amplitudes[i] = 0.0 + 0.0 * I;
    }
    state->amplitudes[index] = 1.0 + 0.0 * I;
}

/**
 * Stampa lo stato quantistico completo del sistema.
 */
void printState(QubitState *state) {
    long long dim = 1LL << state->numQubits;
    for (long long i = 0; i < dim; i++) {
        printf("Stato %lld: %f + %fi | ", i, creal(state->amplitudes[i]),
    cimag(state->amplitudes[i]));
        // Stampa la rappresentazione binaria dell'indice i (da MSB a LSB)
        for (int j = state->numQubits - 1; j >= 0; j--) {
            printf("%d", (i >> j) & 1);
        }
        printf("\n");
    }
}

/**
 * Inizializza lo stato quantistico con tutti i qubit nello stato |0>.
 */
QubitState* initializeState(int numQubits) {
    QubitState *state = (QubitState *)malloc(sizeof(QubitState));
    state->numQubits = numQubits;
    long long dim = 1LL << numQubits;
    state->amplitudes = (double complex *)calloc(dim, sizeof(double complex
    ));

    // Imposta lo stato |0>^N
    state->amplitudes[0] = 1.0 + 0.0 * I;
```

```
67
68      return state;
69  }
70
71  /**
72   * Inizializza il qubit target nello stato |1> mantenendo lo stato degli
        altri qubit.
73   */
74  void initializeSingleQubitToOne(QubitState* state, int target) {
75      long long dim = 1LL << state->numQubits;
76
77      for (long long i = 0; i < dim; i++) {
78          if (((i >> target) & 1) == 0) {
79              long long j = i ^ (1LL << target);
80              state->amplitudes[j] = state->amplitudes[i];
81              state->amplitudes[i] = 0.0 + 0.0 * I;
82          }
83      }
84  }
85
86  /**
87   * Libera la memoria allocata per lo stato quantistico.
88   */
89  void freeState(QubitState *state) {
90      free(state->amplitudes);
91      free(state);
92  }
93
94  /**
95   * Applica un gate a un singolo qubit nello stato quantistico.
96   */
97  void applySingleQubitGate(QubitState *state, int target, double complex
        gate[2][2]) {
98      long long dim = 1LL << state->numQubits;
99      double complex *new_amplitudes = (double complex *)malloc(dim * sizeof(
        double complex));
100
101     for (long long i = 0; i < dim; i++) {
102         int bitValue = (i >> target) & 1;
103         long long j = i ^ (1LL << target);
104
105         if (bitValue == 0) {
106             new_amplitudes[i] = gate[0][0] * state->amplitudes[i] + gate
        [0][1] * state->amplitudes[j];
107         } else {
108             new_amplitudes[i] = gate[1][0] * state->amplitudes[j] + gate
        [1][1] * state->amplitudes[i];
109         }
110     }
111
```

```
112     for (long long i = 0; i < dim; i++) {
113         state->amplitudes[i] = new_amplitudes[i];
114     }
115 
116     free(new_amplitudes);
117 }
118 
119 void applyHadamard(QubitState *state, int target) {
120     double complex H[2][2] = {
121         {1.0 / sqrt(2.0), 1.0 / sqrt(2.0)},
122         {1.0 / sqrt(2.0), -1.0 / sqrt(2.0)}
123     };
124     applySingleQubitGate(state, target, H);
125 }
126 
127 void applyX(QubitState *state, int target) {
128     double complex X[2][2] = {
129         {0, 1},
130         {1, 0}
131     };
132     applySingleQubitGate(state, target, X);
133 }
134 
135 void applyY(QubitState *state, int target) {
136     double complex Y_GATE[2][2] = {
137         {0, -I},
138         {I, 0}
139     };
140     applySingleQubitGate(state, target, Y_GATE);
141 }
142 
143 void applyZ(QubitState *state, int target) {
144     double complex Z[2][2] = {
145         {1, 0},
146         {0, -1}
147     };
148     applySingleQubitGate(state, target, Z);
149 }
150 
151 void applyT(QubitState *state, int target) {
152     double complex T[2][2] = {
153         {1, 0},
154         {0, cexp(I * M_PI / 4.0)}
155     };
156     applySingleQubitGate(state, target, T);
157 }
158 
159 void applyTdag(QubitState *state, int target) {
160     double complex Tdag[2][2] = {
161         {1, 0},
```

```
162         {0, cexp(-I * M_PI / 4.0)}
163     };
164     applySingleQubitGate(state, target, Tdag);
165 }
166 
167 void applyS(QubitState *state, int target) {
168     double complex S[2][2] = {
169         {1, 0},
170         {0, I}
171     };
172     applySingleQubitGate(state, target, S);
173 }
174 
175 /**
176  * Applica un gate CNOT al sistema quantistico.
177  */
178 void applyCNOT(QubitState *state, int control, int target) {
179     long long dim = 1LL << state->numQubits;
180     double complex *new_amplitudes = (double complex *)malloc(dim * sizeof(
    double complex));
181 
182     for (long long i = 0; i < dim; i++) {
183         new_amplitudes[i] = state->amplitudes[i];
184     }
185 
186     for (long long i = 0; i < dim; i++) {
187         int control_bit = (i >> control) & 1;
188         int target_bit = (i >> target) & 1;
189 
190         if (control_bit == 1) {
191             long long j = i ^ (1LL << target);
192             new_amplitudes[i] = state->amplitudes[j];
193             new_amplitudes[j] = state->amplitudes[i];
194         }
195     }
196 
197     for (long long i = 0; i < dim; i++) {
198         state->amplitudes[i] = new_amplitudes[i];
199     }
200 
201     free(new_amplitudes);
202 }
203 
204 /**
205  * Applica un gate Controlled-Z (CZ) al sistema quantistico.
206  */
207 void applyCZ(QubitState *state, int control, int target) {
208     long long dim = 1LL << state->numQubits; // Dimensione dello spazio di
    Hilbert
209 
```

```
210     for (long long i = 0; i < dim; i++) {
211         int control_bit = (i >> control) & 1; // Estrae il valore del qubit
     di controllo
212         int target_bit = (i >> target) & 1;   // Estrae il valore del qubit
     target
213 
214         // Se il qubit di controllo    1 e il target    1, inverti il segno
215         if (control_bit == 1 && target_bit == 1) {
216             state->amplitudes[i] *= -1; // Inversione del segno dell'
    ampiezza
217         }
218     }
219 }
220 
221 
222 void applyCPhaseShift(QubitState *state, int control, int target, double
    complex phase) {
223     long long dim = 1LL << state->numQubits;
224 
225     for (long long i = 0; i < dim; i++) {
226         int control_bit = (i >> control) & 1;
227         int target_bit = (i >> target) & 1;
228 
229         if (control_bit == 1 && target_bit == 1) {
230             state->amplitudes[i] *= phase;
231         }
232     }
233 }
234 
235 /**
236  * Misura il valore di un qubit specificato nello stato quantistico e
    collassa il sistema.
237  */
238 MeasurementResult measure(QubitState *state, int qubit) {
239     long long dim = 1LL << state->numQubits;
240     double prob0 = 0.0;
241 
242     for (long long i = 0; i < dim; i++) {
243         if (((i >> qubit) & 1) == 0) {
244             prob0 += pow(cabs(state->amplitudes[i]), 2);
245         }
246     }
247 
248     double prob1 = 1.0 - prob0;
249     double rand_val = (double)rand() / RAND_MAX;
250     int result = (rand_val < prob0) ? 0 : 1;
251 
252     double scale_factor = 0.0;
253     if (result == 0) {
254         scale_factor = 1.0 / sqrt(prob0);
```

```
255         for (long long i = 0; i < dim; i++) {
256             if (((i >> qubit) & 1) == 1) {
257                 state->amplitudes[i] = 0.0 + 0.0 * I;
258             } else {
259                 state->amplitudes[i] *= scale_factor;
260             }
261         }
262     } else {
263         scale_factor = 1.0 / sqrt(prob1);
264         for (long long i = 0; i < dim; i++) {
265             if (((i >> qubit) & 1) == 0) {
266                 state->amplitudes[i] = 0.0 + 0.0 * I;
267             } else {
268                 state->amplitudes[i] *= scale_factor;
269             }
270         }
271     }
272 
273     MeasurementResult m_result;
274     m_result.prob0 = prob0;
275     m_result.prob1 = prob1;
276     m_result.result = result;
277 
278     return m_result;
279 }
280 
281 /**
282  * Esegue una misura su tutti i qubit del sistema e collassa lo stato.
283  */
284 int* measure_all(QubitState *state) {
285     long long dim = 1LL << state->numQubits;
286     double cumulativeProb = 0.0;
287     double randNum = (double)rand() / RAND_MAX;
288     long long collapse_index = -1;
289 
290     for (long long i = 0; i < dim; i++) {
291         cumulativeProb += pow(cabs(state->amplitudes[i]), 2);
292         if (randNum < cumulativeProb) {
293             collapse_index = i;
294             break;
295         }
296     }
297 
298     int* results = (int*)malloc(state->numQubits * sizeof(int));
299     for (int i = 0; i < state->numQubits; i++) {
300         results[i] = (collapse_index >> i) & 1;
301     }
302 
303     for (long long j = 0; j < dim; j++) {
304         if (j == collapse_index) {
```

```
305             state->amplitudes[j] = 1.0 + 0.0 * I;
306         } else {
307             state->amplitudes[j] = 0.0 + 0.0 * I;
308         }
309     }
310 
311     return results;
312 }
313 
314 QubitAmplitudes getQubitAmplitudes(QubitState* state, int target) {
315     long long dim = 1LL << state->numQubits;
316 
317     QubitAmplitudes result;
318     result.amplitude0 = 0.0 + 0.0 * I;
319     result.amplitude1 = 0.0 + 0.0 * I;
320 
321     for (long long i = 0; i < dim; i++) {
322         if (((i >> target) & 1) == 0) {
323             result.amplitude0 += state->amplitudes[i];
324         } else {
325             result.amplitude1 += state->amplitudes[i];
326         }
327     }
328 
329     return result;
330 }
331 
332 void printQubitAmplitudes(QubitAmplitudes amplitudes) {
333     printf("Ampiezza di |0>: %f + %fi\n", creal(amplitudes.amplitude0),
    cimag(amplitudes.amplitude0));
334     printf("Ampiezza di |1>: %f + %fi\n", creal(amplitudes.amplitude1),
    cimag(amplitudes.amplitude1));
335 }
336 
337 //------------------ 3 qubit gates ----------------------------//
338 
339 void applyToffoli(QubitState* state, int control1, int control2, int target
    ) {
340     applyHadamard(state, target);
341     applyCNOT(state, control2, target);
342     applyTdag(state, target);
343     applyCNOT(state, control1, target);
344     applyT(state, target);
345     applyCNOT(state, control2, target);
346     applyTdag(state, target);
347     applyCNOT(state, control1, target);
348     applyT(state, target);
349     applyHadamard(state, target);
350     //---
351     applyT(state, control2);
```

```
352     applyCNOT(state, control1, control2);
353     applyT(state, control1);
354     applyTdag(state, control2);
355     applyCNOT(state, control1, control2);
356 }
357
358
359
360 void applyFredkin(QubitState* state, int control, int target1, int target2)
        {
361     applyToffoli(state, control, target1, target2);
362     applyCNOT(state, target1, target2);
363     applyToffoli(state, control, target1, target2);
364     applyCNOT(state, target1, target2);
365     applyToffoli(state, control, target1, target2);
366     applyCNOT(state, target1, target2);
367 }
368
369 /**
370  * Applica un gate Controlled-Controlled-Z (CCZ) al sistema quantistico.
371  * Questo gate inverte il segno dello stato target solo se entrambi i qubit
        di controllo sono nello stato |1    .
372  */
373 void applyCCZ(QubitState* state, int control1, int control2, int target) {
374     // Passo 1: Applica Hadamard al qubit target
375     applyHadamard(state, target);
376
377     // Passo 2: Applica il Toffoli gate (CCX)
378     applyToffoli(state, control1, control2, target);
379
380     // Passo 3: Applica Hadamard al qubit target (ripristino)
381     applyHadamard(state, target);
382 }
383
384
385 void _applyCCZ(QubitState* state, int control1, int control2, int target) {
386     applyHadamard(state, target);
387     applyToffoli(state, control1, control2, target);
388     applyHadamard(state, target);
389     applyZ(state, target);
390     applyToffoli(state, control1, control2, target);
391     applyHadamard(state, target);
392 }
393
394 void applyCCY(QubitState* state, int control1, int control2, int target) {
395     applyToffoli(state, control1, control2, target);
396     applyY(state, target);
397     applyToffoli(state, control1, control2, target);
398 }
399
```

```
400 void applyCCPhase(QubitState* state, int control1, int control2, int target
        , double phase) {
401     applyToffoli(state, control1, control2, target);
402     applyPhase(state, target, phase);
403     applyToffoli(state, control1, control2, target);
404 }
405 
406 void applyPhase(QubitState* state, int qubit, double phase) {
407     long long dim = 1LL << state->numQubits;
408 
409     for (long long i = 0; i < dim; i++) {
410         if (((i >> qubit) & 1) == 1) {
411             state->amplitudes[i] *= cexp(I * phase);
412         }
413     }
414 }
```

Listing A.28: Header

# Bibliografia


Scott Aaronson. *Quantum Computing Since Democritus*. Cambridge University Press, 2013.

Frank Arute et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

Rami Barends et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508:500–503, 2014. doi: 10.1038/nature13171.

Giuliano Benenti, Giulio Casati, and Giuliano Strini. *Principles of Quantum Computation and Information*. WORLD SCIENTIFIC, 2004. doi: 10.1142/5528. URL https://www.worldscientific.com/doi/abs/10.1142/5528.

C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973. doi: 10.1147/rd.176.0525.

Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Physical Review Letters*, 70(13):1895–1899, 1993.

Niels Bohr. The quantum postulate and the recent development of atomic theory. *Nature*, 121:580–590, 1928.

David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

David Deutsch. *The Fabric of Reality: The Science of Parallel Universes–and Its Implications*. Penguin, 1997.

David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.

David Deutsch and Chiara Marletto. Constructor theory of information. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2174):20140540, 2015. doi: 10.1098/rspa.2014.0540. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2014.0540.

Paul A. M. Dirac. The quantum theory of the emission and absorption of radiation. *Proceedings of the Royal Society of London. Series A*, 114(767):243–265, 1927.

Paul A. M. Dirac. *The Principles of Quantum Mechanics*. Oxford University Press, 4th edition, 1958.

David P. DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik*, 48(9-11):771–783, 2000.

Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.

Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics, Vol. 3*. Basic Books, 2011.

Luciano Floridi. *The philosophy of information*. Oxford University Press, 2011.

Edward Fredkin and Tommaso Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3-4):219–253, 1982.

James C. Garrison and Raymond Y. Chiao. *Quantum Optics*. Oxford University Press, 2008.

Simon Gay and Ian Mackie. *Semantic Techniques in Quantum Computation*. Cambridge University Press, 2009.

David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. Cambridge University Press, 3rd edition, 2018.

Werner Heisenberg. Über quantentheoretische umdeutung kinematischer und mechanischer beziehungen. *Zeitschrift für Physik*, 33:879–893, 1925.

Werner Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift für Physik*, 43:172–198, 1927.

Max Jammer. *The Conceptual Development of Quantum Mechanics*. American Institute of Physics, 1989.

Lev Davidovich Landau and Evgenii Mikhailovich Lifshitz. *Quantum Mechanics: Non-Relativistic Theory*. Pergamon Press, 1958.

Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

Rolf Landauer. Information is physical. *Physics today*, 44(5):23–29, 1991.

Daniel A. Lidar and Todd A. Brun. *Quantum Error Correction*. Cambridge University Press, 2013.

Seth Lloyd. *Programming the universe: a quantum computer scientist takes on the cosmos*. Vintage, 2006.

C. Marletto. *The Science of Can and Can't: A Physicist's Journey through the Land of Counterfactuals*. Penguin Publishing Group, 2022. ISBN 9780525521945. URL https://books.google.it/books?id=vuaNEAAAQBAJ.

N. David Mermin. *Quantum Computer Science: An Introduction*. Cambridge University Press, 2007.

Eugen Merzbacher. *Quantum Mechanics*. John Wiley & Sons, 3rd edition, 1998.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition edition, 2010.

Jeremy L. O'Brien. Optical quantum computing. *Science*, 318(5856):1567–1570, 2007.

Asher Peres. *Quantum Theory: Concepts and Methods*. Kluwer Academic Publishers, 1995.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

Matthias Reck, Anton Zeilinger, Herbert J. Bernstein, and Philip Bertani. Experimental realization of any discrete unitary operator. *Physical Review Letters*, 73(1):58–61, 1994.

Michael Reed and Barry Simon. *Methods of Modern Mathematical Physics*, volume 1: Functional Analysis. Academic Press, 1980.

J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. Pearson, 2nd edition edition, 2014.

J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. Cambridge University Press, 2nd edition, 2017.

Erwin Schrödinger. Quantisierung als eigenwertproblem. *Annalen der Physik*, 384(4): 361–376, 1926.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

Tommaso Toffoli. Reversible computing. Technical Memo MIT/LCS/TM-151, MIT Lab for Computer Science, 1980.

John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

Wojciech H Zurek. Decoherence, einselection, and the quantum origins of the classical. *Reviews of Modern Physics*, 75(3):715, 2003.