Alma Mater Studiorum · Università di Bologna

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

QUANTUM ERROR CORRECTION E DECODER PER IL TORIC CODE

Relatore:
Dott. Davide Vodola

Presentata da:
Andrea Pari

Anno Accademico 2020/2021

Indice

Sommario

L'argomento principale della tesi è la quantum error correction, in particolare viene esaminato il codice di correzione del toric code e del planar code. Viene inoltre implementato un decoder impiegando l'algoritmo di Edmond che trova il *minimum weight perfect matching* in un grafo.

Dopo un'introduzione sulla meccanica quantistica dei sistemi a due stati e sulla computazione quantistica, viene esaminato l'argomento della quantum error correction nel formalismo degli stabilizer. In seguito viene descritto il codice di correzione del toric code e del planar code e viene illustrata un'implementazione in python di un decoder per il planar code.

Introduzione

L'informatica quantistica è l'insieme delle tecniche di calcolo e del loro studio che utilizzano i principi della meccanica quantistica per memorizzare ed elaborare le informazioni [1]. La ricerca nel campo dell'informatica quantistica e della computazione quantistica è attualmente molto attiva e promettente. Tuttavia la realizzazione di computer quantistici è molto complessa in quanto essi presentano il problema di essere poco affidabili, ovvero sono affetti da un'alta probabilità di errore. La natura probabilistica della meccanica quantistica e la difficoltà della misura degli stati quantistici rende la manipolazione dell'informazione quantistica piuttosto ardua. La branca dell'informatica quantistica che si occupa dello studio degli errori e della protezione dell'informazione quantistica è detta *quantum error correction* [2]. Nel primo capitolo di questa trattazione vengono introdotti i concetti della meccanica quantistica legati all'informatica quantistica: i sistemi a due stati e, in particolare, i qubit. Inoltre vengono descritte le porte logiche quantistiche da un punto di vista matematico, sfruttando il formalismo di Pauli [3]. Il secondo capitolo tratterà la *quantum error correcion*. L'argomento viene introdotto attraverso qualche esempio di semplici codici di correzione quantistici ed in seguito viene analizzato da un punto di vista teorico attraverso il formalismo matematico degli stabilizer [4]. Viene descritto il codice di correzione quantistico del *toric code* [5] e del *planar code* [6]. Inoltre viene trattato il funzionamento del decoder, ovvero l'algoritmo che, dato un errore, automatizza la scelta della correzione. Sia nel caso del toric code che nel caso del planar code, il decoder finora più efficiente è basato su un algoritmo di teoria dei grafi, chiamato algoritmo di Edmond [7, 8]. Il terzo capitolo inizia con una breve introduzione di teoria dei grafi per poi descrivere il funzionamento del decoder. Infine viene proposta un'implementazione di decoder per il planar code e vengono analizzati i risultati della simulazione del funzionamento del decoder.

Capitolo 1

Sistemi a due stati

La computazione quantistica e l'informazione quantistica si basano sul concetto di qubit (bit quantistico). Dal punto di vista della meccanica quantistica il qubit è un esempio di sistema a due stati. Nel primo capitolo di questa trattazione descriveremo il qubit, tratteremo i sistemi a due stati in meccanica quantistica, il formalismo di Pauli e le porte logiche quantistiche.

1.1 Qubit

In fisica quantistica, un sistema a due stati è un sistema che esiste solamente in due stati indipendenti [3]. In generale, ogni sistema che ha energia sufficiente per occupare solamente i due livelli energetici più bassi si comporta come un sistema a due stati. Il qubit rappresenta un esempio di sistema a due stati.

La computazione quantistica e l'informazione quantistica si basano sul concetto di qubit, esso è l'analogo del bit classico in computazione classica.

I qubit, come i bit classici, sono realizzati in sistemi fisici reali, ciononostante in questa trattazione considereremo soltanto l'aspetto teorico del qubit e lo tratteremo come un oggetto matematico astratto, svincolandoci in questo modo dalla realizzazione pratica.

1.1.1 Sistemi costituiti da un singolo qubit

Il qubit [1] è un vettore nello spazio dei ket di un sistema a due stati, $|0\rangle$ e $|1\rangle$. Al contrario dei bit classici, che possono trovarsi solo negli stati 0 e 1, i qubit possono trovarsi in una sovrapposizione di stati quantistici:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \quad \alpha, \beta \in \mathbb{C} \tag{1.1}$$

Lo stato quantistico $|\psi\rangle$ soddisfa la condizione di normalizzazione:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{1.2}$$

Lo stato di un qubit è, quindi, un vettore di modulo unitario nello spazio di Hilbert 2-dimensionale. Gli stati $|0\rangle$ e $|1\rangle$ formano una base ortonormale per questo spazio.

Una differenza fondamentale tra bit e qubit consiste nell'operazione di misura. Mentre è sempre possibile conoscere completamente lo stato di un bit, non è possibile conoscere lo stato quantistico del qubit, ovvero i valori di α e β. In accordo con le leggi della meccanica quantistica, se effettuiamo una misurazione per determinare lo stato di un qubit otteniamo il risultato $|0\rangle$ con probabilità $|\alpha|^2$ e $|1\rangle$ con probabilità $|\beta|^2$.

Un esempio di stato quantistico di un singolo qubit è il seguente:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle. \tag{1.3}$$

Questo stato, una volta effettuata la misurazione per determinarne lo stato, fornisce il risultato $|0\rangle$ con probabilità $\frac{1}{2}$ ed $|1\rangle$ con probabilità $\frac{1}{2}$.

A meno di una fase totale e considerando la condizione di normalizzazione (1.2), lo stato $|\psi\rangle$ può essere riscritto come:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{1.4}$$

Come mostrato in fig. 1.1, gli angoli θ e ϕ individuano un punto sulla sfera di Bloch (vedi sez. 1.2.5), un metodo utile per visualizzare lo stato di un singolo qubit.

1.2 Formalismo di Pauli

Il formalismo di Pauli [3] è il formalismo matematico utilizzato per descrivere i sistemi quantistici a due stati. Esso si fonda sulla scelta di una base particolarmente conveniente per lo spazio degli operatori. Questa base è costituita dagli operatori di Pauli.

1.2.1 Operatori di Pauli

Lo spazio vettoriale dei ket di un sistema a due stati è 2-dimensionale, quello degli operatori è 4-dimensionale, quindi, una base per lo spazio degli operatori consiste di 4 operatori linearmente indipendenti. La base che è conveniente

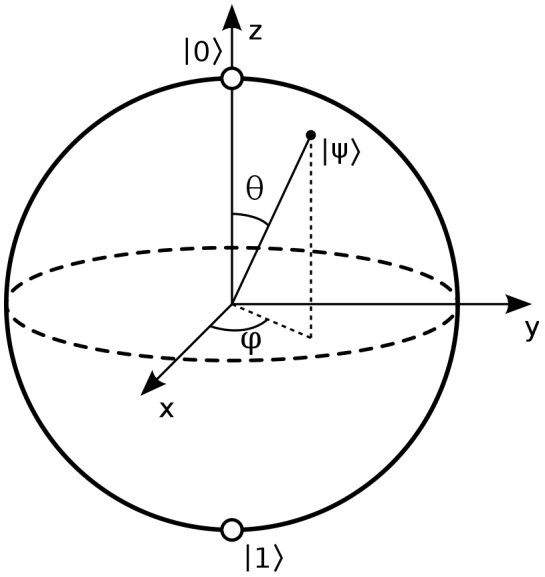

Figura 1.1: Rappresentazione sulla sfera di Bloch dello stato quantistico $|+\rangle$ di un singolo qubit [3].

scegliere è costituita dall'operatore identità e i tre operatori di Pauli. Ogni operatore, $\widehat{A}$, può quindi essere espresso nel modo seguente:

$$\widehat{A} = a_m \widehat{1} + \boldsymbol{a} \cdot \widehat{\boldsymbol{\sigma}}. \tag{1.5}$$

L'operatore di Pauli, $\widehat{\boldsymbol{\sigma}}$, è definito come il vettore di operatori tale che:

$$\boldsymbol{a} \cdot \widehat{\boldsymbol{\sigma}} \boldsymbol{b} \cdot \widehat{\boldsymbol{\sigma}} = \boldsymbol{a} \cdot \boldsymbol{b} \widehat{1} + i \boldsymbol{a} \times \boldsymbol{b} \cdot \widehat{\boldsymbol{\sigma}} \tag{1.6}$$

o equivalentemente

$$\widehat{\boldsymbol{\sigma}} = \widehat{\boldsymbol{\sigma}}^\dagger \tag{1.7}$$

$$[\boldsymbol{a} \cdot \widehat{\boldsymbol{\sigma}}, \boldsymbol{b} \cdot \widehat{\boldsymbol{\sigma}}] = 2i \boldsymbol{a} \times \boldsymbol{b} \cdot \widehat{\boldsymbol{\sigma}} \tag{1.8}$$

$$\boldsymbol{a} \cdot \widehat{\boldsymbol{\sigma}} \boldsymbol{b} \cdot \widehat{\boldsymbol{\sigma}} + \boldsymbol{b} \cdot \widehat{\boldsymbol{\sigma}} \boldsymbol{a} \cdot \widehat{\boldsymbol{\sigma}} = 2 \boldsymbol{a} \cdot \boldsymbol{b} \hat{1} \tag{1.9}$$

L'operatore $\widehat{A}$ espresso come in (1.5) è autoaggiunto se e solo se lo scalare a_m e il vettore $\boldsymbol{a}$ sono reali.

Affrontiamo il problema di trovare gli autovalori e autovettori dell'operatore autoaggiunto $\widehat{A}$. Conviene riscrivere $\widehat{A}$ come:

$$\widehat{A} = a_m \hat{1} + |\boldsymbol{a}| \boldsymbol{a}_1 \cdot \widehat{\boldsymbol{\sigma}} \tag{1.10}$$

Dato che ogni ket non nullo è autoket dell'operatore identità, il problema si riduce a trovare gli autovalori e autovettori di un operatore autoaggiunto

nella forma $\boldsymbol{n} \cdot \widehat{\boldsymbol{\sigma}}$, dove $\boldsymbol{n}$ è un vettore reale di modulo unitario. Lo spettro di $\boldsymbol{n} \cdot \widehat{\boldsymbol{\sigma}}$ è $\lambda_{\pm 1} = \pm 1$ (dimostrazione in [3]).

Denotiamo con $|\boldsymbol{n}, \pm 1\rangle$ gli autovettori di $\boldsymbol{n} \cdot \widehat{\boldsymbol{\sigma}}$ associati rispettivamente agli autovalori $\lambda_{\pm 1} = \pm 1$:

$$\boldsymbol{n} \cdot \widehat{\boldsymbol{\sigma}} \, |\boldsymbol{n}, \pm 1\rangle = |\boldsymbol{n}, \pm 1\rangle \, (\pm 1). \tag{1.11}$$

Tornando al problema iniziale di trovare gli autovalori e autovettori dell'operatore autoaggiunto $\widehat{A}$ si ottiene:

$$\widehat{A} \, |\boldsymbol{n}, \pm 1\rangle = |\boldsymbol{a}_1, \pm 1\rangle \, (a_m \pm |\boldsymbol{a}|). \tag{1.12}$$

1.2.2 Operatori di Pauli rispetto ad una base ortonormale orientata

Una base ortonormale orientata è un insieme di tre vettori, $\boldsymbol{e_i}, i = 1, 2, 3$, che appartengo allo spazio tridimensionale e tali che (l'asterisco indica il complesso coniugato):

$$\boldsymbol{e}_{\boldsymbol{i}}^* = \boldsymbol{e_i} \tag{1.13}$$

$$\boldsymbol{e_i} \cdot \boldsymbol{e_j} = \delta_{ij} \tag{1.14}$$

$$\boldsymbol{e_i} \times \boldsymbol{e_j} = \sum_{k=1}^{3} \epsilon_{ijk} \boldsymbol{e}_k \tag{1.15}$$

Si può rappresentare ogni vettore $\boldsymbol{a}$ rispetto ad una base ortonormale orientata:

$$\boldsymbol{a} = \sum_{i=1}^{3} a_i \boldsymbol{e}_i, \quad a_i = \boldsymbol{e}_i \cdot \boldsymbol{a} \tag{1.16}$$

e, in modo analogo, $\widehat{\boldsymbol{\sigma}}$ si può rappresentare come:

$$\widehat{\boldsymbol{\sigma}} = \sum_{i=1}^{3} \widehat{\sigma_i} \boldsymbol{e}_i, \quad \widehat{\sigma_i} = \boldsymbol{e}_i \cdot \widehat{\boldsymbol{\sigma}}. \tag{1.17}$$

Inoltre per ogni vettore a 3 dimensioni $\boldsymbol{a}$ si ha:

$$\boldsymbol{a} \cdot \widehat{\boldsymbol{\sigma}} = \sum_{i=1}^{3} a_i \widehat{\sigma_i} \tag{1.18}$$

Gli operatori $\widehat{\sigma}_i$ sono autoaggiunti e obbediscono le relazioni (1.6), (1.7), (1.8), (1.9).

Sia $\widehat{\boldsymbol{\sigma}}$ un operatore di Pauli. Scegliamo una base ortonormale orientata nello spazio a tre dimensioni. Sia $|\boldsymbol{e}_3, \pm 1\rangle$ una base ortonormale di autoket di $\widehat{\sigma}_3$:

$$\widehat{\sigma}_3 |\boldsymbol{e}_3, \pm 1\rangle = |\boldsymbol{e}_3, \pm 1\rangle (\pm 1) \tag{1.19}$$

Scegliendo opportunamente la fase dei ket $|\boldsymbol{e}_3, \pm 1\rangle$, abbiamo:

$$\widehat{\sigma}_1 = |\boldsymbol{e}_3, -1\rangle \langle \boldsymbol{e}_3, +1| + |\boldsymbol{e}_3, +1\rangle \langle \boldsymbol{e}_3, -1| \tag{1.20}$$

$$\widehat{\sigma}_2 = i |\boldsymbol{e}_3, -1\rangle \langle \boldsymbol{e}_3, +1| - i |\boldsymbol{e}_3, +1\rangle \langle \boldsymbol{e}_3, -1| \tag{1.21}$$

$$\widehat{\sigma}_3 = |\boldsymbol{e}_3, +1\rangle \langle \boldsymbol{e}_3, +1| - |\boldsymbol{e}_3, -1\rangle \langle \boldsymbol{e}_3, -1| . \tag{1.22}$$

Riscrivendo le equazioni precedenti in forma matriciale otteniamo le matrici ortonormali di Pauli:

$$\widehat{\sigma}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \widehat{\sigma}_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \widehat{\sigma}_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.23}$$

1.2.3 Operatori di Pauli rispetto ad una base sferica orientata

Definiamo una base sferica orientata come un insieme di tre vettori dello spazio complesso a tre dimensioni, $\boldsymbol{e}_\alpha, \alpha = -1, 0, +1$, tali che (l'asterisco indica il complesso coniugato):

$$\boldsymbol{e}_\alpha^* = \boldsymbol{e}_{-\alpha} \tag{1.24}$$

$$\boldsymbol{e}_\alpha \cdot \boldsymbol{e}_\beta = 2^{|\alpha|} \delta_{-\alpha\beta} \tag{1.25}$$

$$\boldsymbol{e}_\alpha \times \boldsymbol{e}_\beta = -i\delta_{\alpha 0}\beta \boldsymbol{e}_\beta + i\delta_{\beta 0}\alpha \boldsymbol{e}_\alpha - i\delta_{-\alpha\beta}(\alpha - \beta)\boldsymbol{e}_0. \tag{1.26}$$

La base ortonormale e la base sferica sono in corrispondenza uno a uno secondo le relazioni seguenti:

$$\boldsymbol{e}_0 = \boldsymbol{e}_3, \quad \boldsymbol{e}_{\pm 1} = \boldsymbol{e}_1 \pm i\boldsymbol{e}_2 \tag{1.27}$$

$$\boldsymbol{e}_1 = \frac{1}{2}(\boldsymbol{e}_{+1} + \boldsymbol{e}_{-1}), \quad \boldsymbol{e}_2 = \frac{1}{2i}(\boldsymbol{e}_{+1} - \boldsymbol{e}_{-1}), \quad \boldsymbol{e}_3 = \boldsymbol{e}_0 \tag{1.28}$$

Scegliamo una base ortonormale orientata nello spazio a tre dimensioni. Sia $|\boldsymbol{e}_3, \pm 1\rangle$ una base ortonormale di autoket di $\widehat{\sigma}_0$:

$$\widehat{\sigma}_0 |\boldsymbol{e}_0, \pm 1\rangle = |\boldsymbol{e}_0, \pm 1\rangle (\pm 1) \tag{1.29}$$

Scegliendo opportunamente la fase dei ket $|\boldsymbol{e}_0, \pm 1\rangle$, abbiamo:

$$\widehat{\sigma}_0 = |\boldsymbol{e}_0, +1\rangle \langle \boldsymbol{e}_0, +1| - |\boldsymbol{e}_0, -1\rangle \langle \boldsymbol{e}_0, -1| \tag{1.30}$$

$$\widehat{\sigma}_{\pm 1} = 2 |\boldsymbol{e}_0, \pm 1\rangle \langle \boldsymbol{e}_0, \mp 1| . \tag{1.31}$$

Riscrivendo le equazioni precedenti in forma matriciale otteniamo le matrici sferiche di Pauli:

$$\widehat{\sigma}_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \widehat{\sigma}_{+1} = \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix}, \quad \widehat{\sigma}_{-1} = \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix}. \tag{1.32}$$

1.2.4 Esponenziale di operatori di Pauli

L'algebra degli operatori di Pauli si rivela molto utile nello lo studio dei sistemi fisici a due stati in meccanica quantistica. Ne è un esempio la formula per il calcolo dell'esponenziale di un operatore di Pauli [3]:

$$\exp\left(i\frac{\theta}{2}\boldsymbol{k}\cdot\widehat{\boldsymbol{\sigma}}\right) = \cos\left(\frac{\theta}{2}\right)\widehat{1} + i\sin\left(\frac{\theta}{2}\right)\boldsymbol{k}\cdot\widehat{\boldsymbol{\sigma}} \tag{1.33}$$

dove θ è un angolo e $\boldsymbol{k}$ un vettore unitario.

La formula dell'esponenziale di Pauli può essere usata per ruotare un operatore di Pauli su un altro operatore di Pauli. Siano $\mathbf{u}$, $\mathbf{v}$ due vettori unitari, sia θ l'angolo compreso tra essi e $\mathbf{k}$ il vettore unitario perpendicolare ad entrambi, allora abbiamo:

$$\exp\left(i\frac{\theta}{2}\boldsymbol{k}\cdot\widehat{\boldsymbol{\sigma}}\right)\boldsymbol{u}\cdot\widehat{\boldsymbol{\sigma}}\exp\left(-i\frac{\theta}{2}\boldsymbol{k}\cdot\widehat{\boldsymbol{\sigma}}\right) = \boldsymbol{v}\cdot\widehat{\boldsymbol{\sigma}} \tag{1.34}$$

ed inoltre

$$|\boldsymbol{v}, \pm 1\rangle = \exp\left(i\frac{\theta}{2}\boldsymbol{k}\cdot\widehat{\boldsymbol{\sigma}}\right)|\boldsymbol{u}, \pm\mathbf{1}\rangle\,. \tag{1.35}$$

1.2.5 Sfera di Bloch

Sia $\boldsymbol{e}_\alpha$ una base sferica orientata tale che gli operatori di Pauli $\widehat{\sigma}_\alpha$ sono dati da (1.28) e sia $\boldsymbol{n}$ un vettore unitario. Applicando l'equazione (1.31) otteniamo:

$$|\boldsymbol{n}, +1\rangle = |\boldsymbol{e}_0, +1\rangle\cos\left(\frac{\theta}{2}\right) + |\boldsymbol{e}_0, -1\rangle\sin\left(\frac{\theta}{2}\right)\exp(i\phi) \tag{1.36}$$

$$|\boldsymbol{n}, -1\rangle = -|\boldsymbol{e}_0, +1\rangle\sin\left(\frac{\theta}{2}\right)\exp(-i\phi) + |\boldsymbol{e}_0, -1\rangle\cos\left(\frac{\theta}{2}\right) \tag{1.37}$$

dove l'angolo θ e la phase ϕ sono definite da:

$$\cos\theta = n_0, \quad \exp(\pm i\phi) = \frac{n_{\pm 1}}{|n_{\pm 1}|}. \tag{1.38}$$

Sia $|0, \pm 1\rangle$ una base ortonormale. I ket $|0, \pm 1\rangle$ sono nella forma $|\boldsymbol{n}, \pm 1\rangle$ a meno di fattori di fase. Quindi, la coppia di vettori ortonormali $|0, \pm 1\rangle$ sono in corrispondenza uno a uno con i punti $\boldsymbol{n}$ della sfera unitaria nello spazio tridimensionale, la sfera di Bloch. Gli angoli θ e ϕ sono gli angoli che corrispondono al vettore $\boldsymbol{n}$.

1.3 Porte logiche quantistiche

Le porte logiche quantistiche [1], in analogia con le porte logiche classiche, hanno lo scopo di manipolare l'informazione. Le porte logiche quantistiche agiscono sempre in modo lineare, questo comportamento è una proprietà generale della meccanica quantistica. Per la linearità dell'azione delle porte quantistiche è possibile rappresentarle in forma matriciale. La condizione di normalizzazione degli stati quantistici impone che la matrice, U, che rappresenta una generica porta quantistica, sia unitaria, ovvero: $U^\dagger U = I$. La condizione di unitarietà è l'unica richiesta necessaria affinché una matrice rappresenti una valida porta quantistica.

Consideriamo le porte quantistiche ad un qubit, esse possono essere rappresentate da matrici 2×2. Nel caso classico l'unica porta logica non triviale ad un bit in entrata è la porta NOT, mentre, nel caso quantistico, esistono infinite possibili porte quantistiche ad una entrata. L'equivalente quantistico della porta NOT agisce trasformando lo stato $|0\rangle$ in $|1\rangle$ e viceversa:

$$\alpha\,|0\rangle + \beta\,|1\rangle \longrightarrow \alpha\,|1\rangle + \beta\,|0\rangle \tag{1.39}$$

La matrice, X, che rappresenta la porta NOT è definita nel modo seguente:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{1.40}$$

Lo stato di un qubit è rappresentato in forma vettoriale nel modo seguente:

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \begin{bmatrix} |0\rangle & |1\rangle \end{bmatrix}. \tag{1.41}$$

Applicare la porta NOT, X, allo stato quantistico, $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, corrisponde al prodotto tra la matrice X e il vettore $|\psi\rangle$:

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}. \tag{1.42}$$

Omettiamo, per alleggerire la notazione, il vettore riga $[|0\rangle\,|1\rangle]$.

Altri esempi rilevanti sono la porta di Pauli Z, la porta di Pauli Y, la porta di Hadamard H, la porta S e la porta T (anche detta porta $\pi/8$). La porta Z non modifica $|0\rangle$ e scambia il segno di $|1\rangle$:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.43}$$

Le porte di Y, H, S, T sono rappresentate dalle seguenti matrici:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} T = \begin{bmatrix} 1 & 0 \\ 0 & \exp \frac{i\pi}{4} \end{bmatrix} \tag{1.44}$$

Consideriamo, ora, le porte logiche quantistiche a più qubit. L'esempio più rilevante di porta logica quantistica a più qubit è la porta *controlled-NOT* (CNOT). La porta CNOT riceve in entrata due qubit: il qubit di controllo a e il qubit di target b. La porta agisce nel modo seguente: se il qubit di controllo è 0, il qubit di target non viene modificato, se il qubit di controllo è 1, il qubit di target viene invertito. La matrice che rappresenta la porta CNOT è la seguente:

$$U_{CN} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.45}$$

La porta CNOT insieme con le porte a singolo qubit sono porte universali, ovvero ogni porta logica a più qubit può essere costruita utilizzando solamente porte CNOT e porte a singolo qubit (dimostrazione in [1]).

La base ortonormale dello spazio di Hilbert 2-dimensionale costituita dai vettori $|0\rangle$ e $|1\rangle$, che abbiamo utilizzato finora per descrivere lo spazio degli stati di un qubit è detta base computazionale. La meccanica quantistica permette di effettuare misurazioni in basi diverse dalla base computazionale. In generale, data una qualunque base $|a\rangle, |b\rangle$ per un qubit, è possibile esprimere uno stato qualunque come una combinazione lineare $\alpha |a\rangle + \beta |b\rangle$.

1.3.1 Porte logiche quantistiche e operatori di Pauli

Nella base computazionale gli operatori di Pauli assumono la forma (sez. 1.2):

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{1.46}$$

Gli operatori di Pauli sono autoaggiunti e unitari e ciò implica rispettivamente che definiscono delle misure e delle porte logiche.

Riassumiamo le proprietà degli operatori di Pauli:

- Hermitiani
- Unitari
- Self-inverse $X^2 = Y^2 = Z^2 = I$

- Due operatori di Pauli qualunque commutano o anticommutano
- Hanno autovalori ± 1
- Formano una base per le matrici 2×2
- Includendo i prefattori $\pm 1, \pm i$, gli operatori di Pauli formano un gruppo. L'intero gruppo contiene anche operatori non hermitiani, noi ci limiteremo a considerare solo i membri hermitiani del gruppo degli operatori di Pauli.

Capitolo 2

Quantum error correction

Fino ad ora abbiamo considerato sistemi quantistici isolati, ovvero sistemi che non hanno interazioni col mondo esterno. I sistemi reali, però, non sono mai perfettamente isolati dall'ambiente esterno. Essi interagiscono inevitabilmente con l'ambiente. Queste interazioni non volute costituiscono il rumore nell'informazione quantistica [1]. Per costruire sistemi informatici quantistici è essenziale comprendere e controllare il rumore nell'informazione.

In questo capitolo, dopo una breve introduzione sull'argomento della correzione degli errori, tratteremo il formalismo degli stabilizer ed il toric code.

2.1 Introduzione

Al fine di proteggere un messaggio dagli errori dovuti al rumore si codifica il messaggio aggiungendo informazioni ridondanti, in modo tale che, nel caso in cui parte dell'informazione contenuta nel messaggio codificato venga corrotta, ci sia abbastanza ridondanza di informazione da rendere possibile decodificare il messaggio codificato e recuperare il messaggio originale.

Consideriamo il seguente esempio [1]. Supponiamo di dover inviare un bit attraverso un canale di comunicazione classico che presenta del rumore. L'effetto del rumore è di invertire il bit con probabilità p. Con probabilità $1-p$ il bit è trasmesso senza errore. Un modo per proteggere il bit dal rumore è di rimpiazzarlo con tre copie di se stesso:

$$0 \longrightarrow 000, \quad 1 \longrightarrow 111. \tag{2.1}$$

Le stringhe di bit 000 e 111 sono chiamate *0 logico* e *1 logico*. Supponiamo che la stringa di bit in uscita dal canale sia 001. Nella condizione molto comune che p sia una probabilità piccola, è molto più probabile che soltanto

il terzo bit sia stato invertito e quindi che il messaggio originale fosse 000 rispetto al caso in cui siano stati invertiti i primi due bit e il messaggio originale sia 111. Il codice appena descritto funziona a condizione che avvenga l'inversione di al più un bit. Questo codice è detto *repetition code*. Per proteggere stati quantistici dall'errore dovuto al rumore esistono codici che si basano su principi analoghi a quelli appena descritti. Ci sono però importanti differenze tra l'informazione classica e quella quantistica che vanno tenute in considerazione nello sviluppo dei codici per la correzione di errori quantistici. In particolare:

- **No cloning:** la meccanica quantistica proibisce la clonazione di uno stato. È impossibile, quindi, implementare meccanicamente il repetition code duplicando lo stato iniziale.
- **Errori continui:** su un qubit può verificarsi un continuo di errori, contrariamente al caso classico nel quale l'unico errore possibile su un singolo bit è il bit flip.
- **La misura distrugge lo stato quantistico:** la misurazione dello stato quantistico in uscita dal canale distrugge lo stato e rende impossibile la decodifica.

Vediamo ora come risolvere questi problemi considerando due esempi.

2.2 Codice bit flip

Il codice bit flip protegge dall'errore di bit flip che è l'operazione rappresentata dall'operatore X. Consideriamo una situazione analoga a quella dell'esempio precedente. Un qubit attraversa un canale di comunicazione che presenta del rumore tale che con probabilità p il qubit viene invertito. Questo tipo di canale viene detto canale di bit flip. Supponiamo di codificare lo stato del singolo qubit in tre qubit:

$$|0\rangle \longrightarrow |0_L\rangle = |000\rangle\,, \quad |1\rangle \longrightarrow |1_L\rangle = |111\rangle\,. \tag{2.2}$$

Quindi lo stato quantistico è:

$$|\psi_L\rangle = a\,|0_L\rangle + b\,|1_L\rangle \tag{2.3}$$

Supponiamo che si sia verificato un bit flip su al più un qubit. Lo stato originale può essere ripristinato in due passaggi [1]:

1. **Rilevamento dell'errore.** Effettuiamo una misura che rileva quale errore si è verificato. Il risultato della misura si chiama sindrome di errore (*error syndrome*). Per il canale di bit flip esistono quattro sindromi di errore possibili che corrispondono ai quattro operatori di proiezione:

   - $P_0 = |000\rangle\langle 000| + |111\rangle\langle 111|$ Nessun errore
   - $P_1 = |100\rangle\langle 100| + |011\rangle\langle 011|$ Bit flip sul primo qubit
   - $P_2 = |010\rangle\langle 010| + |101\rangle\langle 101|$ Bit flip sul secondo qubit
   - $P_3 = |001\rangle\langle 001| + |110\rangle\langle 110|$ Bit flip sul terzo qubit

   Supponiamo che si sia verificato un bit flip sul secondo qubit, lo stato in uscita dal canale è:

   $$|\psi_L\rangle = a\,|010\rangle + b\,|101\rangle\,. \tag{2.4}$$

   Applicando l'operatore di proiezione P_2 otteniamo

   $$\langle\psi|\,P_2\,|\psi\rangle = 1. \tag{2.5}$$

   La sindrome di errore relativa al secondo qubit è uguale ad 1 quindi abbiamo rilevato un bit flip sul secondo qubit. La sindrome di errore contiene solamente l'informazione sul verificarsi o meno dell'errore, non contiene alcuna informazione sullo stato quantistico. Infatti, per ottenere informazioni sull'identità di uno stato quantistico sarebbe necessario perturbarne lo stato.

   Un altro modo per rilevare l'errore è misurare la parità dello stato codificato $|\psi_L\rangle$ [2]. Gli operatori che misurano la parità dello stato $|\psi_L\rangle$ sono: $S_1 = Z \otimes Z \otimes I$, $S_2 = Z \otimes I \otimes Z$ e $S_3 = I \otimes Z \otimes Z$. Se, misurando tali operatori, si ottengono autovalori $+1$ lo stato è senza errori.

   Se è avvenuto un errore di bit flip, per esempio:

   $$|\psi_L\rangle = a\,|010\rangle + b\,|101\rangle\,, \tag{2.6}$$

   la misura degli operatori S_1 e S_3 restituiranno l'autovalore -1 mentre la misura dell'operatore S_2 restituirà $+1$.

   Errori differenti sullo stato $|\psi_L\rangle$ restituiscono diverse combinazioni di autovalori risultanti dalla misura degli operatori:

Error	$Z \otimes Z \otimes I$	$Z \otimes I \otimes Z$	$I \otimes Z \otimes Z$
$X \otimes I \otimes I$	-1	-1	+1
$I \otimes X \otimes I$	-1	+1	-1
$I \otimes I \otimes X$	+1	-1	-1

Figura 2.1: Tipi di errore di bit flip associati alle combinazioni di autovalori risultanti dalla misura degli operatori. Notiamo che i risultati in tabella corrispondono al fatto che l'errore commuti $(+1)$ o anticommuti (-1) con l'operatore di misura della parità [2].

2. **Recupero:** Usiamo l'informazione della sindrome di errore per recuperare lo stato corretto. Nel nostro caso la sindrome di errore indica un bit flip sul secondo qubit, quindi invertiamo lo stesso qubit e otteniamo lo stato iniziale.

$$|\psi_{output}\rangle = a\,|010\rangle + b\,|101\rangle \longrightarrow |\psi_{corretto}\rangle = a\,|000\rangle + b\,|111\rangle\,. \quad (2.7)$$

Questa procedura di correzione degli errori funziona a patto che il bit flip avvenga su al più un singolo qubit, in analogia con l'esempio riguardante il caso classico.

2.3 Codice phase flip

Consideriamo ora il caso dell'errore di phase flip, rappresentato dall'operatore Z. Quando il qubit attraversa il canale di phase flip, si verifica l'inversione della fase dello stato quantistico con probabilità p.

$$Z\,(a\,|0\rangle + b\,|1\rangle) = a\,|0\rangle - b\,|1\rangle \quad (2.8)$$

Supponiamo di lavorare nella base

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \quad (2.9)$$

In questa base l'operatore Z si comporta come l'operatore di bit flip X nella base computazionale, infatti:

$$Z\,|+\rangle = Z\frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle \quad (2.10)$$

$$Z\,|-\rangle = Z\frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \quad (2.11)$$

Ridefiniamo quindi gli stati logici

$$|0_L\rangle = |+++\rangle\,, \quad |1_L\rangle = |---\rangle\,. \tag{2.12}$$

In questo modo le procedure di rilevamento dell'errore e di correzione del phase flip rispetto alla base $|+\rangle$, $|-\rangle$ vengono effettuate esattamente come nel caso del bit flip in base computazionale. Il cambio di base viene effettuato applicando allo stato quantistico la porta di Hadamard H.

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.13}$$

$$H\,|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \tag{2.14}$$

$$H\,|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle \tag{2.15}$$

Per tornare alla base computazionale è sufficiente riapplicare la porta di Hadamard, in quanto le porte logiche quantistiche sono rappresentate da operatori unitari.

2.4 Operatori logici sugli stati codificati

Vogliamo definire le porte logiche che agiscono sugli stati codificati. A questo scopo è sufficiente identificare gli operatori X e Z che agiscono sugli stati codificati poichè l'operatore Y si ottiene da $Y = iXZ$ e gli operatori X, Y, Z e I formano una base completa di operatori.

L'operatore di bit flip per lo stato codificato è l'operatore $\bar{X} = X \otimes X \otimes X$ infatti:

$$(X \otimes X \otimes X)\,|000\rangle = |111\rangle \quad (X \otimes X \otimes X)\,|111\rangle = |000\rangle\,. \tag{2.16}$$

L'operatore Z per lo stato codificato è $\bar{Z} = Z \otimes I \otimes I$, infatti:

$$(Z \otimes I \otimes I)\,|000\rangle = |000\rangle \quad (Z \otimes I \otimes I)\,|111\rangle = -\,|111\rangle\,. \tag{2.17}$$

L'operatore $\bar{Z}$ non è unico, avremmo potuto scegliere anche $\bar{Z} = I \otimes Z \otimes I$ oppure $\bar{Z} = I \otimes I \otimes Z$.

Notiamo, inoltre, che gli operatori logici codificati $\bar{X}$ e $\bar{Z}$ soddisfano l'algebra degli operatori di Pauli.

2.5 Distanza del codice

La distanza di un codice quantistico è il peso minimo di ogni operatore logico codificato diverso dall'identità. Il peso di un operatore è il numero di qubit su cui agisce in maniera non triviale (E.g. $Z \otimes Z \otimes I$ ha peso 2).

Nel caso del repetition code a 3 qubit, il peso minimo dell'operatore codificato $\bar{X} = X \otimes X \otimes X$ è 3. Il codice, può quindi rilevare fino 2 errori X. Il peso minimo dell'operatore codificato $\bar{Z} = Z \otimes I \otimes I$ è 1 perciò il codice non è in grado di rilevare errori Z.

2.6 Formalismo degli stabilizer

Il formalismo degli stabilizer [2, 4] è una tecnica per definire e studiare i codici quantistici per la correzione degli errori. Si basa sull'utilizzo degli operatori di Pauli al posto di lavorare direttamente con i vettori ket degli stati quantistici.

Introduciamo l'argomento con il seguente esempio. Consideriamo lo stato quantistico di Bell di due qubit:

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{2.18}$$

Questo stato soddisfa le identità

$$X_1 X_2 |\psi\rangle = |\psi\rangle\,, \quad Z_1 Z_2 |\psi\rangle = |\psi\rangle\,. \tag{2.19}$$

Lo stato è, quindi, autostato di X_1X_2 e Z_1Z_2 con autovalore $+1$. Diciamo che lo stato $|\psi\rangle$ è stabilizzato dagli operatori X_1X_2 e Z_1Z_2, inoltre $|\psi\rangle$ è l'unico stato ad essere stabilizzato dai due operatori appena citati. L'idea alla base del formalismo degli stabilizer è che molti stati quantistici sono più facilmente descrivibili mediante gli operatori che li stabilizzano rispetto allo stato stesso in forma esplicita. Molti codici quantistici possono essere descritti in maniera più semplice e compatta sfruttando questo formalismo e per di più gli errori sui qubit e sulle operazioni su di essi sono descritti più facilmente nel formalismo degli stabilizer.

Descriveremo gli stabilizer utilizzando la teoria dei gruppi.

Gli operatori di Pauli, includendo i prefattori ± 1 e $\pm i$, formano un gruppo rispetto all'operazione di prodotto di matrice, detto gruppo di Pauli.

Chiamiamo G_n il gruppo di Pauli su n qubits. Sia S un sottogruppo di G_n, sia V_S l'insieme degli stati di n qubit che sono fissati da ogni elemento di S. V_S è lo spazio vettoriale stabilizzato da S ed S si dice stabilizzatore

dello spazio V_S poichè ogni elemento di V_S è stabile rispetto all'azione degli elementi di S.

Per descrivere un gruppo si utilizzano i suoi generatori. Un insieme di elementi $g_1, ..., g_n$ appartenenti ad un gruppo G si dice che genera il gruppo G se ogni elemento di G può essere scritto come un prodotto di elementi dell'insieme $g_1, ..., g_n$ e si scrive $G = \langle g_1, ..., g_n \rangle$.

Vediamo il caso semplice del repetition code con $n = 3$ e $S = \{I, Z_1Z_2, Z_2Z_3, Z_1Z_3\}$. Il sottogruppo S è generato da $S = \{Z_1Z_2, Z_2Z_3\}$, infatti $Z_1Z_3 = (Z_1Z_2)(Z_2Z_3)$ e $I = (Z_1Z_2)^2$. Cerchiamo lo spazio V_S stabilizzato da S. È sufficiente considerare i generatori di S in quanto se un vettore di V_S è stabilizzato dai generatori di S è automaticamente stabilizzato anche dal prodotto degli elementi di S. Il sottospazio fissato da Z_1Z_2 è generato da $|000\rangle$, $|001\rangle$, $|110\rangle$, $|111\rangle$. Il sottospazio fissato da Z_2Z_3 è generato da $|000\rangle$, $|100\rangle$, $|011\rangle$, $|111\rangle$. Gli unici elementi comuni ai due insiemi sono $|000\rangle$ e $|111\rangle$. Quindi, V_S è generato da questi ultimi due elementi.

Non tutti i sottogruppi S degli operatori di Pauli possono essere usati come stabilizer per uno spazio vettoriale non triviale. È semplice individuare due condizioni necessarie che devono essere soddisfatte da S affinchè stabilizzi uno spazio non triviale:

- **Gli elementi di S commutano.** Siano g_1, g_2 elementi di S. Se g_1 e g_2 non commutano, abbiamo che se $g_1g_2 |\psi\rangle = g_2g_1 |\psi\rangle = |\psi\rangle$ allora $|\psi\rangle = 0$.
- **$-I$ non è un elemento di S.** Se $(-I) |\psi\rangle = |\psi\rangle$ allora $|\psi\rangle = 0$.

Si dimostra che queste condizioni sono anche sufficienti affinché lo spazio V_S stabilizzato da S sia non triviale [1].

Un codice di correzione quantistico che può essere definito nel formalismo degli stabilizer è chiamato *stabilizer code*. Gli stabilizer code sono definiti specificando due insiemi di operatori: i generatori di stabilizer e gli operatori logici codificati.

2.6.1 Gruppo degli stabilizer e generatori

Consideriamo un codice di correzione quantistico con stati di base $|\psi_j\rangle$. Il gruppo di stabilizer [2] è l'insieme di operatori di Pauli P che lasciano tutti gli stati di base del codice invariati:

$$P_k |\psi_j\rangle = |\psi_j\rangle . \tag{2.20}$$

Gli operatori stabilizer commutano e formano un gruppo abeliano che è un sottogruppo del gruppo degli operatoti di Pauli.

Un gruppo G può essere descritto da un insieme di generatori g_j, dove $j = 1, ..., m$. Nel caso di un gruppo abeliano di operatori self-inverse, ogni elemento $g \in G$ può essere scritto:

$$g = \prod_j g_j^{a_j} \tag{2.21}$$

dove $a_j \in 0, 1$. Ogni elemento può essere espresso in termini della stringa di bit $a_1 a_2 ... a_m$. Diciamo che un insieme di generatori è indipendente se l'unica soluzione di

$$\prod_j g_j^{a_j} = I \tag{2.22}$$

è $a_j = 0$ per tutti i j. Questa condizione di indipendenza implica che nessun generatore può essere scritto come prodotto di altri generatori e che qualunque operatore g è descritto dall'equazione (2.21) mediante un'unica stringa di bit. Inoltre, quest'ultimo fatto implica che l'ordine di G è 2^m, dove m è il numero di generatori.

Per un codice stabilizer, il numero di generatori m, il numero di qubit logici codificati k e il numero di qubit n sono correlati dalla formula:

$$m = n - k. \tag{2.23}$$

Esempio del repetition code

Consideriamo l'esempio del repetition code. Gli stati logici di base per il repetition code a 3 qubit sono:

$$|0_L\rangle = |000\rangle \quad |1_L\rangle = |111\rangle . \tag{2.24}$$

Per questo codice, $n = 3$, $k = 1$, quindi $m = 2$. L'ordine del gruppo è $2^2 = 4$. Infatti, i seguenti 4 operatori lasciano invariati gli stati di base:

$$ZZI \quad ZIZ \quad IZZ \quad III. \tag{2.25}$$

Tutti gli operatori commutano e presi due operatori qualunque tra ZZI, ZIZ e IZZ formano un insieme di generatori indipendenti per il gruppo.

2.6.2 Rilevazione degli errori nel formalismo degli stabilizer

Consideriamo l'esempio precedente. Gli operatori stabilizer del repetition code a 3 qubit (ad eccezione dell'operatore identità) sono gli stessi operatori

che vengono misurati per rilevare gli errori di bit-flip. Gli operatori di Pauli sono operatori hermitiani e perciò definiscono degli osservabili misurabili. Possiamo reinterpretare la definizione di stabilizer nel modo seguente: esso è l'insieme delle misurazioni che restituiscono autovalori $+1$ nel caso di stati senza errore e autovalori -1 se uno stato ha un errore rilevabile. Quindi, la rilevazione degli errori nei codici stabilizer avviene misurando gli operatori stabilizer. Per le proprietà dei gruppi è sufficiente misurare un insieme indipendente di generatori di stabilizer.

2.6.3 Operatori logici codificati

In sez. 2.4 abbiamo derivato gli operatori logici per il repetition code a 3 qubit. Abbiamo notato che l'operatore codificato $\bar{Z}$ ha diverse forme alternative equivalenti: $Z \otimes I \otimes I$, $I \otimes Z \otimes I$, $I \otimes I \otimes Z$.
Il formalismo degli stabilizer fornisce una semplice caratterizzazione di questa equivalenza. Considero lo stato $|\psi\rangle$ nello spazio del codice, un operatore logico codificato $\bar{L}$ e un elemento del gruppo di stabilizer S_j. Dato che $S_j |\psi\rangle = |\psi\rangle$, si può mostrare che $\bar{L}S_j |\psi\rangle = \bar{L} |\psi\rangle$, quindi gli operatori $\bar{L}S_j$ agiscono identicamente ad $\bar{L}$ sugli stati nello spazio del codice. Ciò implica che esistono 2^m operatori equivalenti per ogni operatore $\bar{L}$.

Gli operatori logici codificati devono commutare con ogni elemento del gruppo degli stabilizer, poiché se un operatore di Pauli anticommuta con un elemento dello stabilizer non può lasciare lo spazio del codice invariante.
L'insieme di operatori che commutano con ogni elemento dello stabilizer è chiamato *centralizer* di questo sottogruppo.

Gli operatori logici codificati, oltre a dover commutare con tutti gli elementi dello stabilizer, devono soddisfare le relazioni di commutazione degli operatori di Pauli che rappresentano.

2.6.4 Relazione tra errore e sindrome

In sez. 2.2 abbiamo visto che, nel caso del codice bit flip, il risultato della misura dello stabilizer era $+1$ quando l'errore commutava con l'operatore di misura e -1 quando anticommutava. È possibile generalizzare questa relazione ad ogni tipo di errore [2].

Sia $|\psi\rangle$ lo stato prima dell'errore. Sia E l'operatore di Pauli che rappresenta l'errore e S un generatore dello stabilizer.

Se E commuta con S, lo stato dopo l'errore, $E |\psi\rangle$, è un autostato di S con autovalore $+1$.

$$E |\psi\rangle = ES |\psi\rangle = SE |\psi\rangle \tag{2.26}$$

Se E anticommuta con S:

$$E\,|\psi\rangle = ES\,|\psi\rangle = -SE\,|\psi\rangle\,, \tag{2.27}$$

lo stato dopo l'errore, $E\,|\psi\rangle$, è un autostato di S con autovalore -1.

La relazione tra errori e misure degli stabilizer non è $1-1$. Dato un errore E e un elemento dello stabilizer S, gli errori E ed SE porteranno alla misura dello stesso risultato. Questo fenomeno è chiamato *degenerazione del codice*. La scelta dell'operatore di correzione dell'errore più appropriato non è banale. Gli algoritmi che automatizzano questa scelta sono detti *decoder* [2].

2.7 Toric code

Il toric code è l'esempio più semplice di codice topologico. È stato ideato da Alexei Kitaev nel 1996 [5].

Il toric code [2] è definito come un reticolo quadrato con contorni periodici. Consideriamo un reticolo $L \times L$ quadrato, avvolto attorno ad un toro (fig. 2.2), tale che il contorno destro è identificato con quello sinistro e il contorno in alto è identificato con quello in basso.

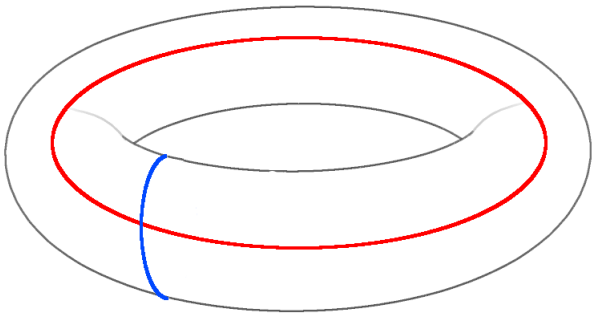

Figura 2.2: Il toro [2].

Il reticolo è costituito da spigoli, vertici (punti di incontro degli spigoli) e placchette (tasselli quadrati individuati da un insieme di spigoli), come mostrato in fig. 2.3. Associamo un qubit ad ogni spigolo sul reticolo. Su un reticolo $L \times L$ con condizioni al contorno periodiche ci sono $2L^2$ spigoli. Il toric code ha, quindi, $n = 2L^2$ qubit fisici.

Per ogni placchetta definiamo un'*operatore placchetta* che consiste in un operatore dato dal prodotto tensore di operatori di Pauli Z che agiscono sui 4 qubit al contorno della placchetta (fig. 2.4). Su tutti gli altri qubit agiscono operatori identità. Su un reticolo $L \times L$ ci sono L^2 placchette.

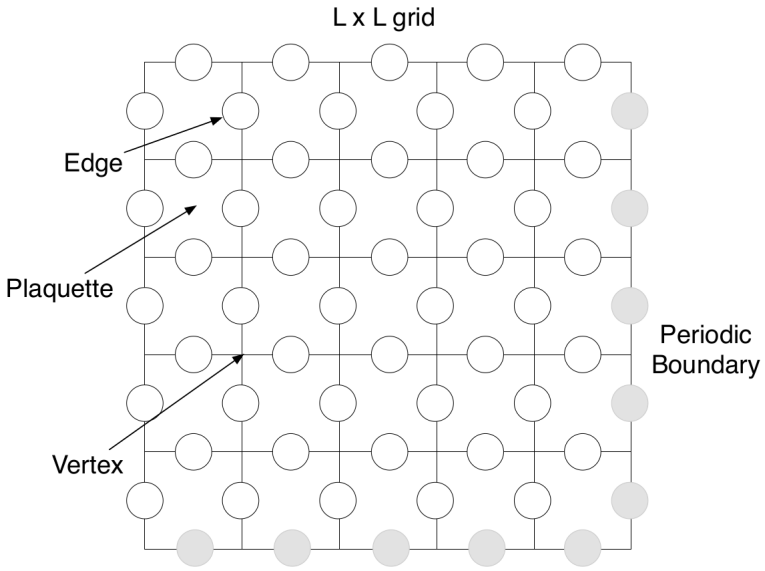

Figura 2.3: Il toric code è definito in termini di un reticolo $L \times L$ con condizioni al contorno periodiche. Ogni cerchio inserito su ogni spigolo rappresenta un qubit [2].

Per ogni vertice definiamo un'operatore vertice che consiste in un operatore dato dal prodotto tensore di operatori di Pauli X che agiscono su 4 qubits adiacenti a un vertice (fig. 2.4). Su tutti gli altri qubit agiscono operatori identità. Su un reticolo $L \times L$ ci sono L^2 vertici.

Ricordiamo che un codice di stabilizer di n qubit che codifica k qubit è definito specificando $m = n - k$ generatori indipendenti e gli operatori logici codificati $\bar{X}$ e $\bar{Z}$.

2.7.1 Commutazione di operatori

Come abbiamo visto in sez. 2.6, i generatori devono commutare. Gli operatori dello stesso tipo, ovviamente, commutano tra loro. Gli operatori associati a placchette e vertici che non sono vicini commutano dato che agiscono su qubit diversi. Ci resta da analizzare il caso di operatori che agiscono sugli stessi qubits. Un vertice all'angolo di una placchetta è adiacente a due spigoli della placchetta, quindi gli operatori di placchetta e di vertice agiscono contemporaneamente su due qubits. Nonostante X e Z anticommutino, i prodotti $X \otimes X$ e $Z \otimes Z$ commutano. Abbiamo mostrato che anche gli operatori placchetta e vertice vicini commutano e quindi tutti gli operatori placchetta e vertice commutano tra loro.

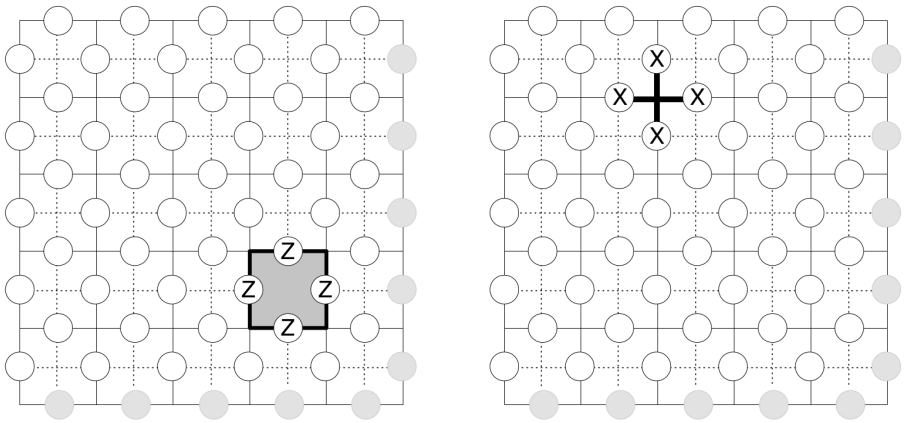

Figura 2.4: Esempio di operatore placchetta e operatore vertice [2].

2.7.2 Moltiplicazione di operatori placchetta

Studiamo, in primo luogo, la moltiplicazione di operatori placchetta. Consideriamo due placchette adiacenti. Sul qubit in comune tra le due agiscono due operatori Z. Se moltiplichiamo i due operatori placchetta, i due operatori Z sul bordo in comune si moltiplicano e diventano l'operatore identità, $Z^2 = I$. Quindi, l'operatore risultante dalla moltiplicazione di due operatori placchetta adiacenti sono 6 operatori Z attorno al contorno delle due placchette.

In generale, qualunque coppia di placchette ha un singolo spigolo in comune o nessuno. Moltiplicando un insieme di placchette qualunque, gli operatori Z sugli spigoli condivisi diventano l'operatore identità e quindi l'operatore risultante dal prodotto consiste nel bordo complessivo delle placchette (fig. 2.5).

Consideriamo il prodotto di tutte le placchette del reticolo. Il suo bordo non esiste poiché le placchette ricoprono tutto il reticolo avvolto attorno al toro. Quindi, il prodotto di tutti gli operatori placchetta è l'identità. Ciò implica che l'insieme di tutte le placchette non è un insieme di operatori indipendenti. Per ottenere un insieme di generatori indipendenti è sufficiente rimuovere una placchetta qualunque. Abbiamo, quindi, $L^2 - 1$ operatori placchetta indipendenti.

Notiamo, inoltre, che il contorno di un insieme di placchette consiste sempre di cicli chiusi di operatori Z.

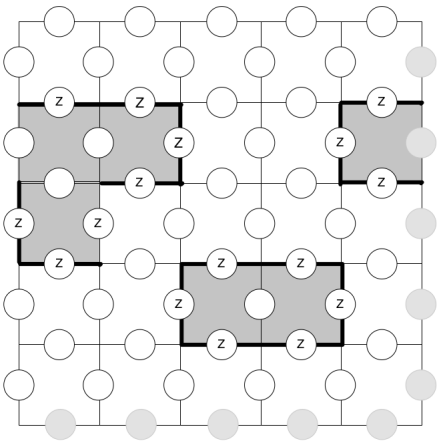

Figura 2.5: L'operatore risultante dalla moltiplicazione di un insieme di placchette consisterà di operatori Z che agiscono sul contorno delle placchette unite [2].

2.7.3 Moltiplicazione di operatori vertice

Chiamiamo il reticolo iniziale, che abbiamo considerato fino ad ora, reticolo primario. Costruiamo un secondo reticolo, il reticolo duale, traslando di metà cella verso il basso e metà cella verso destra il reticolo primario. Il reticolo duale ha le stesse dimensioni e condizioni al bordo del reticolo primario, ma ogni placchetta nel reticolo primario è diventata un vertice in quello duale e viceversa.

La dualità del reticolo è uno strumento potente nello studio del toric code poiché possiamo scegliere quale reticolo è più adatto per ogni situazione. Nel caso della moltiplicazione di operatori è più semplice la rappresentazione a placchette.

In generale, il duale di un reticolo si ottiene scambiando placchette e vertici e riorientando gli spigoli. Solitamente la struttura del reticolo cambia. Nel caso del reticolo quadrato, il reticolo primario ha la stessa struttura del reticolo duale, si dice perciò self-dual. Questa proprietà rende il toric code su reticolo quadrato adatto a definire codici che proteggono equivalentemente dagli errori di tipo X e Z.

Sfruttando la dualità del reticolo, gli operatori vertice possono essere definiti come il contorno del corrispondente operatore placchetta nel reticolo duale. Analogamente al caso delle placchette, la moltiplicazione di operatori vertice può essere descritta in termini di contorno complessivo delle placchette nel reticolo duale. Per le stesse ragioni presentate nel caso degli operatori placchetta, il prodotto di tutti gli operatori vertice è l'identità, rimuoven-

do un elemento otteniamo un insieme di generatori indipendenti di $L^2 - 1$ elementi.

2.7.4 Qubits codificati

Nel toric code ci sono $n = 2L^2$ qubit fisici, $L^2 - 1$ operatori placchetta e $L^2 - 1$ operatori vertice, quindi il numero di operatori è $m = 2(L^2 - 1)$. Dall'eq. 2.23 abbiamo che il numero di qubit codificati è $k = n - m = 2$. Per completare la descrizione del codice non ci resta che identificare gli operatori logici codificati $\bar{Z}_1$, $\bar{X}_1$, $\bar{Z}_2$, $\bar{X}_2$.

Come abbiamo visto in sez. 2.6.3, gli operatori logici codificati devono commutare con tutti gli elementi dello stabilizer, non possono essere elementi dello stabilizer e devono soddisfare le relazioni di commutazione degli operatori che rappresentano, in questo caso X e Z.

Operatore codificato $\bar{Z}_1$

Cerchiamo un prodotto di operatori Z che commuta con ogni stabilizer e che non sia uno stabilizer. Se consideriamo una stringa di operatori Z, indipendentemente dalla forma, essa anticommuterà con gli operatori vertice agli estremi di essa. La soluzione, quindi, è di scegliere una stringa di operatori che formano un ciclo chiuso. Identifichiamo con $\bar{Z}_1$ l'operatore codificato, associato al primo qubit, formato da un ciclo chiuso di operatori Z, come mostrato in fig. 2.6.

Operatore codificato $\bar{X}_1$

Cerchiamo ora l'operatore $\bar{X}_1$. $\bar{X}_1$ deve commutare con ogni stabilizer e anticommutare con $\bar{Z}_1$. Per la proprietà di dualità del reticolo, un ciclo chiuso di operatori X commuta con ogni stabilizer (fig. 2.6). Inoltre, l'operatore $\bar{X}_1$ anticommuta con $\bar{Z}_1$ poiché condividono soltanto un qubit sul quale agiscono in modo non triviale.

Operatori codificati $\bar{Z}_2$ e $\bar{X}_2$

Se ruotiamo i cicli di operatori corrispondenti a $\bar{Z}_1$ e $\bar{X}_1$ di 90 gradi, otteniamo due nuovi operatori che commutano con $\bar{Z}_1$ e $\bar{X}_1$, commutano con lo stabilizer e anticommutano tra loro. Questi sono gli operatori logici $\bar{Z}_2$ e $\bar{X}_2$ (fig. 2.6).

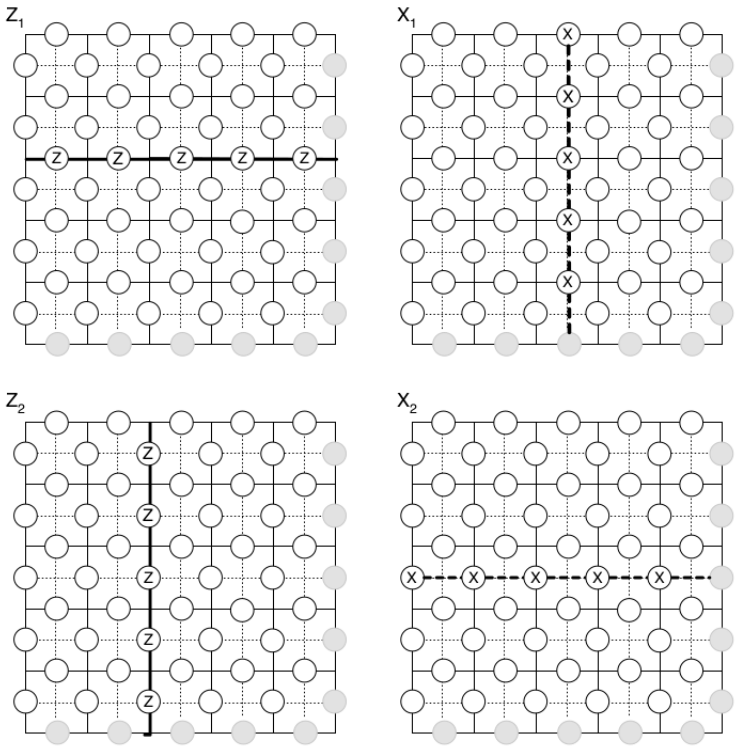

Figura 2.6: Operatori logici codificati $\bar{Z}_1$, $\bar{X}_1$, $\bar{Z}_2$ e $\bar{X}_2$ per i due qubit codificati nel toric code [2].

2.7.5 Rilevazione dell'errore nel toric code

Consideriamo un errore Z su un singolo qubit. Tutti gli operatori placchetta e gli operatori vertice che non agiscono sul quel qubit commutano con esso. Gli unici stabilizer che anticommutano con l'operatore errore sono gli operatori vertice adiacenti ad esso. Quindi, questo singolo errore ha invertito il valore delle misure di due stabilizer.

Consideriamo ora due errori Z su qubit adiacenti. L'operatore vertice tra i due errori commuta con l'operatore errore, poichè $X \otimes X$ commuta con $Z \otimes Z$. Gli unici operatori vertice i cui valori sono invertiti sono quelli agli estremi della stringa di errori. Ciò è valido in generale.

La rilevazione degli errori di tipo X avviene in maniera del tutto analoga. I due tipi di errore agiscono indipendentemente su diversi tipi di stabilizer, possiamo, quindi, considerarli due processi separati di correzione degli errori. Da ora in poi, tratteremo solamente gli errori di tipo Z.

Per correggere gli errori, applichiamo l'inverso dell'operatore errore. Nel

caso di errori di Pauli, per la proprietà di self-inverse, applichiamo l'operatore stesso. Il problema principale nella correzione degli errori è l'identificazione dell'operatore correzione da applicare data una determinata sindrome di errore. Come abbiamo già visto, la relazione tra sindrome ed errore non è $1-1$. Due operatori di Pauli E ed E' porteranno alla stessa sindrome di errore se $E = E'L$ dove L è un operatore qualunque che commuta con lo stabilizer. Se L appartiene allo stabilizer del codice, il prodotto $EE' = L$ è nello stabilizer. Un errore E seguito da una correzione E' si combinano e generano un operatore dello stabilizer. Quindi, E' corregge E.

Gli operatori logici codificati commutano anch'essi con lo stabilizer. Perciò, dato l'errore E, gli errori $E\bar{Z}_1$, $E\bar{Z}_2$, $E\bar{Z}_1\bar{Z}_2$ portano tutti alla stessa sindrome di errore.

Per una data sindrome, gli operatori errore che possono averla generata si dividono in quattro classi di equivalenza. Dato un singolo operatore E, le quattro classi di equivalenza possono essere ottenute moltiplicando E con gli operatori I, $E\bar{Z}_1$, $E\bar{Z}_2$, $E\bar{Z}_1\bar{Z}_2$ e da qualunque elemento nello stabilizer formato soltanto da prodotti di operatori placchetta. Scegliere la correzione più appropriata è un problema complesso, la cui soluzione dipende dal tipo di modello di errore adottato. Nel nostro caso tratteremo il modello *indipendent noise model*. In questo modello, gli errori sui singoli qubit avvengono indipendentemente in accordo con la seguente distribuzione:

- **I (no error):** $1-p^2$
- **X:** $p(1-p)$
- **Y:** p^2
- **Z:** $p(1-p)$

Questo processo è equivalente a una combinazione di due processi indipendenti:

- **I (no error):** $1-p$
- **X:** p

e

- **I (no error):** $1-p$
- **Z:** p.

In questo modello di errore, i due tipi di errore sono indipendenti, quindi è possibile studiare i due tipi di sindromi (vertice e placchetta) indipendentemente.

2.7.6 Code threshold

La distanza del codice nel caso del toric code su un reticolo $L \times L$ è L. Aumentando L ci si aspetterebbe un aumento della robustezza del codice, ma considerando il modello *indipendent noise model*, aumentare L significa aumentare il numero di errori e ciò rende più complessa la scelta di un operatore di correzione adatto. Quando la probabilità di errore è abbastanza bassa, aumentare L aumenterà la probabilità di correzione dell'errore. Quando la probabilità di errore è alta, aumentare L abbasserà la probabilità di correzione dell'errore. Il punto di transizione tra questi due effetti è detto *code threshold*. Il valore di threshold dipende dal tipo di algoritmo usato dal decoder. Il threshold per il miglior decoder possibile è detto *optimal threshold*. Per il toric code nel modello di *indipendent noise model* il threshold dell'errore ottimale è stato stimato essere 11% [9].

2.8 Decoder ottimale

Al fine di correggere l'errore E, applichiamo l'operatore C che appartiene alla stessa classe di equivalenza di E, ovvero $C = ES$ con S elemento qualunque dello stabilizer. Se applichiamo l'operatore C che appartiene ad un'altra classe di equivalenza, $C = E\bar{U}S$ con $\bar{U} \in \{\bar{Z}_1, \bar{Z}_2, \bar{Z}_1\bar{Z}_2\}$, l'informazione verrà corrotta da un errore logico non rilevabile di tipo $\bar{Z}_1$, $\bar{Z}_2$ o $\bar{Z}_1\bar{Z}_2$. Se il decoder fornisce un operatore di correzione in un'altra classe di equivalenza allora la correzione è fallita.

Un decoder ottimale è tale per cui, analizzando le probabilità di tutti i possibili errori che possono aver generato una data sindrome, fornisce l'operatore di correzione nella classe di equivalenza più probabile.

Un decoder ottimale non è efficiente, il numero dei possibili operatori di errore scala esponenzialmente con L. Non è, quindi, il decoder utilizzato in pratica. Tuttavia, il decoder ottimale è utile come confronto con altri decoder efficienti e ci permette di calcolare il limite superiore del threshold dell'errore. Si dimostra [9] che, adottando il modello di errore *indipendent noise model*, il threshold ottimale dell'errore corrisponde a una probabilità di errore dell'11%.

Il decoder più utilizzato ed efficiente per il toric code si basa sull' algoritmo del *Minimum Weight Perfect Matching* (MWPM). Si stima numericamente che il threshold di errore per questo tipo di decoder sia del 10.3% [2]. Il decoder MWPM sarà l'argomento centrale del prossimo capitolo.

Capitolo 3

Quantum error correction nel toric code

Il decoder più utilizzato ed efficiente per il toric code è basato sull'algoritmo di Minumum Weight Perfect Matching (MWPM). In questo capitolo tratteremo i concetti fondamentali della teoria dei grafi al fine di descrivere efficacemente il problema del MWPM. Inoltre verrà esposto un esempio di implementazione in python di un decoder basato sull'algoritmo MWPM.

3.1 Teoria dei grafi

In matematica, la teoria dei grafi [7] è la disciplina che si occupa dello studio dei grafi, strutture matematiche discrete che permettono di modellizzare le relazioni tra coppie di oggetti.

Definizione 1 (Grafo, vertici, nodi)**.** *Un grafo (fig. 3.1) è una coppia $G = (V, E)$ di insiemi tali che $E \subseteq [V]^2$; quindi, gli elementi di E sono sottoinsiemi di V costituiti da due elementi. Gli elementi di V sono i vertici (o nodi) del grafo G. Gli elementi di E sono gli spigoli (edges).*

Definizione 2 (Grafo pesato)**.** *Un* ***grafo pesato (weighted graph)*** *è un grafo nel quale un numero (il peso) è associato ad ogni spigolo.*

Definizione 3 (Vertici incidenti)**.** *Un vertice v è incidente con uno spigolo e se $v \in e$ ed inoltre e è uno spigolo di v. I due vertici incidenti con uno spigolo sono i suoi estremi. Uno spigolo x, y viene solitamente denotato con xy o yx. Se gli spigoli non hanno un orientazione allora si dice che il grafo è non orientato (o indiretto), altrimenti si dice che è orientato (o diretto).*

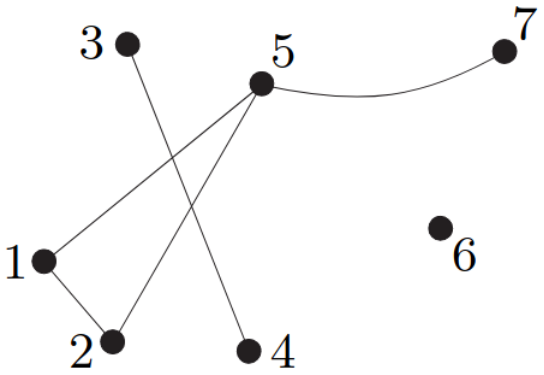

Figura 3.1: Esempio di grafo. $V = \{1, \ldots, 7\}$ $E = \{\{1,2\},\{1,5\},\{2,5\},\{3,4\},\{5,7\}$ [7].

Definizione 4 (Grafo diretto/indiretto)**.** *Se gli spigoli non hanno un'orientazione allora si dice che il grafo è* ***non orientato (o indiretto)****, altrimenti si dice che è* ***orientato (o diretto)****.*

Definizione 5 (Accoppiamento)**.** *Un insieme M di spigoli indipendenti in un grafo $G = (V, E)$ è chiamato* ***accoppiamento*** *o* ***matching****. M è un accoppiamento per $U \subseteq V$ se ogni vertice in U è incidente con uno spigolo in M. I vertici in U sono detti* ***accoppiati*** *e i vertici non incidenti con alcun vertice di M sono* ***non accoppiati****.*

Definizione 6 (Accoppiamento massimale)**.** *Un* ***accoppiamento massimale (maximal matching)*** *è un accoppiamento M di un grafo G che non è un sottoinsieme di nessun altro accoppiamento. Un accoppiamento M di un grafo G è massimale se ogni spigolo in G ha un'intersezione non vuota con almeno uno spigolo in M (fig. 3.2).*

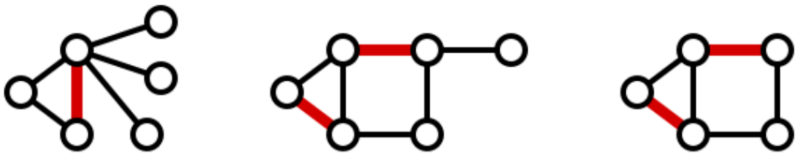

Figura 3.2: Esempi di accoppiamenti massimali [10].

Definizione 7 (Accoppiamento massimo)**.** *Un* ***accoppiamento massimo (maximum matching)*** *è un accoppiamento che contiene il massimo numero possibile di spigoli. Può esistere più di un accoppiamento massimo. Ogni*

accoppiamento massimo è massimale, ma non ogni accoppiamento massimale è un accoppiamento massimo (fig. 3.3).

Definizione 8 (Accoppiamento perfetto)**.** *Un* ***accoppiamento perfetto (perfect matching)*** *è un accoppiamento che abbina tutti i vertici di un grafo. Un accoppiamento è perfetto se ogni vertice del grafo è incidente a uno spigolo dell'accoppiamento. Ogni accoppiamento perfetto è massimo e massimale. Un grafo può contenere un accoppiamento perfetto solo se ha un numero pari di vertici.*

Definizione 9 (Accoppiamento perfetto con peso minimo)**.** *Un accoppiamento perfetto con peso minimo (minimum weight perfect matching) è un accoppiamento perfetto tale che la somma dei pesi associati ad ogni spigolo è minima.*

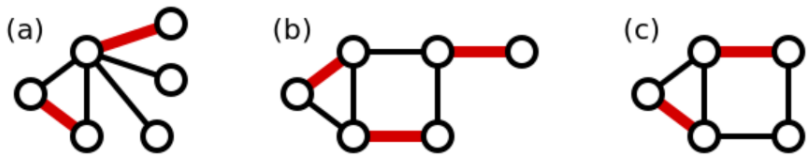

Figura 3.3: Esempi di accoppiamenti massimi. L'accoppiamento (b) è un accoppiamento perfetto [10].

Algoritmo di Edmond

L'algoritmo di Edmond è un algoritmo per trovare un accoppiamento perfetto con peso minimo in un grafo pesato indiretto. Fu sviluppato negli anni '60 da Jack Edmond. L'efficienza dell'algoritmo è $O(V^3)$, dove V è il numero di vertici.

3.2 Decoder per il toric code

Dopo aver introdotto alcune nozioni di base di teoria dei grafi e descritto il problema del MWPM, vediamo il funzionamento del decoder sul toric code.

Consideriamo solo errori Z, poichè adottando il modello di errore *uncorrelated noise model*, possiamo trattare i due tipi di errore indipendentemente ed in modo analogo. Consideriamo n qubit sui quali può avvenire un errore Z con probabilità p. La probabilità di ottenere un errore di peso m è $p^m(1-p)^{N-m}$, dove N è il numero totale di qubit. Quindi, l'errore più

probabile è quello col peso minimo. L'algoritmo di Edmond calcola l'errore con peso minimo data una sindrome.

Una sindrome di errore consiste nella misura dell'autovalore dello stabilizer uguale a -1 agli estremi della stringa di errori. Per correggere gli errori dobbiamo trovare il percorso più probabile che congiunge un autovalore -1 ad un altro autovalore -1 in modo da annullare la sindrome. Costruiamo un grafo costituito da tutti gli operatori vertice con autovalore -1 e associamo ad ogni spigolo un peso uguale alla distanza di Manhattan tra i vertici nel reticolo.

L'operatore di correzione col peso minimo corrisponde al matching perfetto con peso minimo del grafo (fig. 3.4).

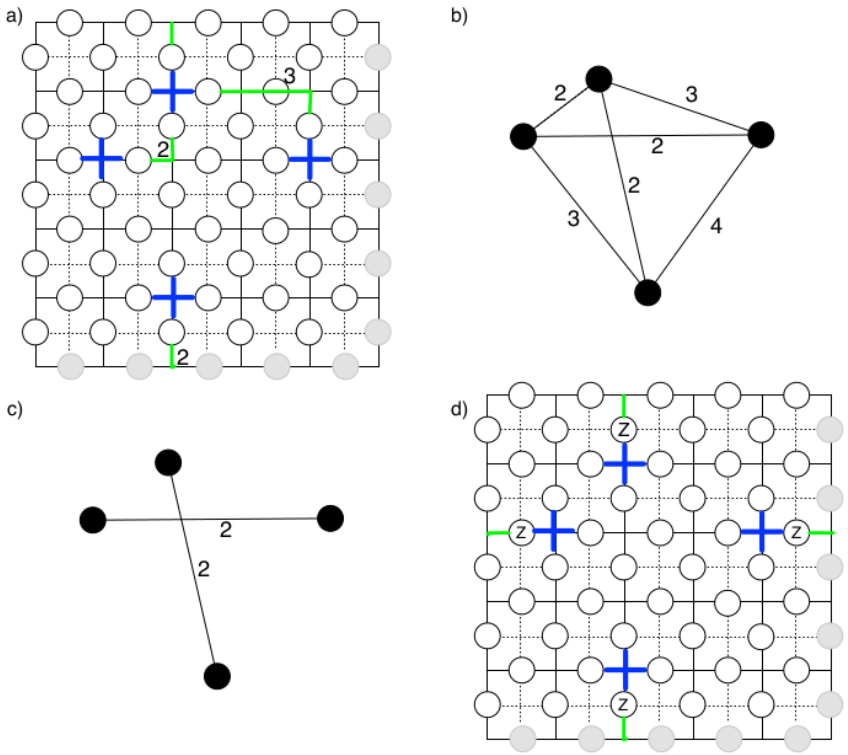

Figura 3.4: Applicazione dell'algoritmo MWPM per trovare l'operatore di correzione con peso minimo. In fig. a) i vertici in blu sono i vertici con autovalore -1, gli spigoli verdi rappresentano le distanze di Manhattan minime tra i vertici in blu. In fig. b) è rappresentato un grafo dove i nodi corrispondono ai vertici in blu della fig. a) e i pesi degli spigoli corrispondono alla distanza di manhattain tra i vertici in blu. In fig. c) è rappresentato il matching perfetto con peso minimo del grafo in fig. b). In fig. d) è rappresentato l'operatore di correzione corrispondente al matching [2].

3.3 Implementazione del decoder

Per semplificare la simulazione, consideriamo un codice planare (o *planar code*). In questo modo evitiamo la complessità dei contorni periodici. Il funzionamento del decoder è il medesimo rispetto al caso del toric code.

3.3.1 Planar code

Un codice planare [11] codifica un solo qubit ed ha la struttura in fig. 3.5. A differenza del toric code, il planar code presenta due tipi di contorno differenti: il *rough boundary* costituito da una fila di placchette incomplete che presentano solo tre spigoli e lo *smooth boundary* che invece è costitutito da placchette complete.

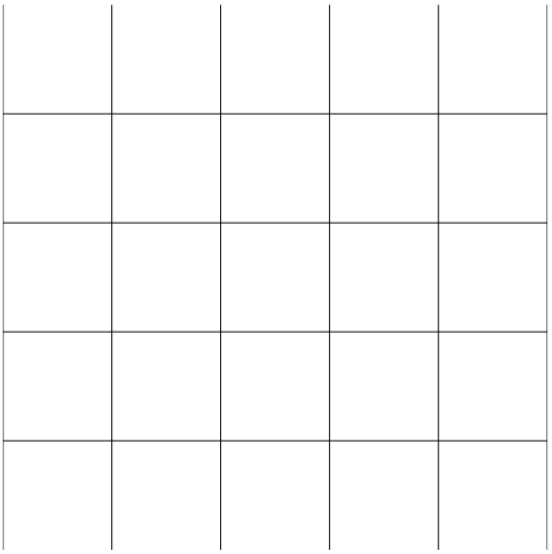

Figura 3.5: Esempio di struttura di codice planare che codifica un singolo qubit. I contorni laterali sono detto *smooth boundaries*. I contorni in alto e in basso sono detti *rough boundaries* [11].

Verifichiamo che questa struttura permette di codificare un solo qubit. Nel caso in fig. 3.5 il reticolo rappresentato ha $n = 50$ qubit, 24 vertici, 15 placchette e 10 placchette incomplete (che comunque definiscono dei generatori). Quindi in totale ci sono $m = 49$ generatori di stabilizer. Il codice codifica $k = n - m = 1$ qubit. Questo calcolo è valido per ogni planar code. Il qubit logico ha i corrispondenti operatori logici $\bar{X}$ e $\bar{Z}$, come mostrato in fig.

3.6. Analogamente al caso del toric code gli operatori logici si estendono da un contorno a quello opposto, indipendentemente dal percorso che seguono.

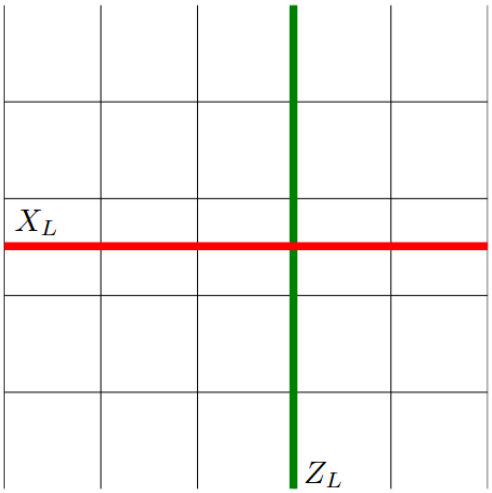

Figura 3.6: I due operatori logici del planar code. L'operatore Z logico è una stringa di operatori Z che collegano i due *rough boundaries*. L'operatore X logico è una sequenza di operatori X che collegano i due *smooth boundaries*. In figura sono rappresentati come due stringhe di operatori nel reticolo duale [11].

3.3.2 Implementazione del decoder sul planar code

L'implementazione del decoder è in python e sfrutta le seguenti librerie:

- networkx: per la gestione e visualizzazione dei grafi [12],
- numpy: per il calcolo numerico,
- matplotlib: per i grafici.

Il programma si suddivide in quattro fasi:

1. la generazione casuale di errori,
2. il calcolo degli autovalori,
3. l'applicazione dell'algoritmo di Edmond,
4. la verifica della correzione.

Generazione casuale di errori

Per ogni qubit viene simulato un errore con probabilità p di avvenire. Consideriamo in questa implementazione del decoder soltanto errori di tipo Z. In fig. 3.7 e 3.8 (sinistra) è rappresentato il grafo degli errori, dove gli spigoli rossi rappresentano i qubit con errore e quelli verdi senza errore.

Calcolo degli autovalori

Come illustrato in sez. 2.7.5, una stringa di errori Z implica che la misura degli operatori vertice X agli estremi della stringa restituisca l' autovalore -1. In base a questo fatto, calcoliamo gli autovalori degli operatori vertice del grafo degli errori. Il grafo che rappresenta gli autovalori è rappresentato in fig. 3.7 e 3.8 (centro). L'obiettivo del decoder è calcolare un operatore di correzione conoscendo solamente gli autovalori degli operatori vertice.

Applicazione dell'algoritmo di Edmond

Vogliamo calcolare la stringa di operatori più probabile (ovvero con peso minimo) che accoppi i vertici con autovalore -1. Costruiamo, quindi, un grafo che ha: come nodi i vertici con autovalore -1 e come spigoli tutti i possibili accoppiamenti dei nodi. Ad ogni spigolo è associato un peso uguale alla distanza di Manhattan dei vertici nel grafo degli autovalori. Su questo grafo applichiamo l'algoritmo di Edmond per trovare l'accoppiamento perfetto più probabile. L'operatore di correzione sarà ottenuto collegando gli accoppiamenti attraverso il percorso più breve possibile sul grafo degli autovalori. L'operatore di correzione è rappresentato in fig. 3.7 e 3.8 (destra).

Verifica della correzione

L'ultima fase consiste nel verificare l'esito della correzione. Per fare ciò costruiamo una matrice, chiamata rappresentazione binaria [6], che ha una riga per ogni generatore di stabilizer e un numero di colonne uguale al numero totale di qubit. Indicizziamo i qubit (ricordando che ogni spigolo ospita un qubit) come in fig. 3.7 e 3.8. Ogni riga della rappresentazione binaria rappresenterà un generatore di stabilizer e avrà un 1 in corrispondenza degli operatori che fanno parte del generatore e uno 0 su tutti gli altri. Dato che i generatori sono linearmente indipendenti anche le righe della rappresentazione saranno linearmente indipendenti e la matrice avrà rango massimo pari al numero di righe. Costruiamo il vettore che rappresenta l'errore ($\boldsymbol{v}_{err}$), che avrà 1 nelle posizioni corrispondenti ai qubit sui quali è avvenuto un errore e 0 sugli altri. Analogamente costruiamo il vettore che rappresenta la

correzione ($\boldsymbol{v}_{corr}$), che avrà 1 nelle posizioni corrispondenti agli operatori di correzione e 0 sugli altri. Infine sommiamo in modulo 2 il vettore dell'errore ed il vettore della correzione, chiamo il vettore risultante $\boldsymbol{v}_{err-corr}$:

$$\boldsymbol{v}_{err-corr} = (\boldsymbol{v}_{err} + \boldsymbol{v}_{corr}) \mod 2. \tag{3.1}$$

Se il vettore $\boldsymbol{v}_{err-corr}$ e i vettori riga della rappresentazione lineare sono linearmente indipendenti, la correzione sarà fallita poiché significa che è avvenuto un errore logico che non è possibile correggere. Nel caso in cui il vettore $\boldsymbol{v}_{err-corr}$ e i vettori riga della rappresentazione lineare siano linearmente dipendenti allora la correzione ha avuto esito positivo, poiché l'errore e la correzione sono equivalenti a uno o più generatori.

Sfruttando l'algoritmo di Gauss in modulo 2 calcoliamo il rango della matrice ottenuta dalla rappresentazione binaria alla quale è stata aggiunta la riga ulteriore del vettore $\boldsymbol{v}_{err-corr}$. Se il rango di questa matrice è uguale al rango della rappresentazione binaria, il vettore $\boldsymbol{v}_{err-corr}$ è linearmente dipendente con i vettori della rappresentazione quindi la correzione ha avuto esito positivo. Nel caso in cui il rango sia maggiore di una unità allora il vettore $\boldsymbol{v}_{err-corr}$ è linearmente indipendente e quindi la correzione è fallita.

Stima del valore di threshold dell'errore

Effettuiamo un numero $N_s = 1000$ di simulazioni di correzione per ogni valore di probabilità di errore p variabile tra 0% e 30% con uno step di 1%. Ripetiamo questo procedimento per 4 dimensioni del codice planare: $n = 162, 200, 242, 288$ (n indica il numero totale di qubit). Rappresentiamo nei grafici in fig. 3.9 e 3.10 la percentuale di correzioni con esito positivo rispetto al totale delle simulazioni in funzione della percentuale di errore p.

Le curve ottenute sono tra di loro compatibili, tuttavia presentano delle fluttuazioni statistiche dovute al basso numero di simulazioni di correzione effettuate. Per via dell'alto costo computazionale della simulazione non è stato possibile generale un campione di dati più grande. Dai grafici emerge che all'aumentare della probabilità di errore p, la probabilità di correzione scende rapidamente. In particolare per $p \approx 7\%$ è circa 80%, per $p \approx 10\%$ è attorno al 60% e scende al 10% per $p \approx 20\%$.

La stima del valore di threshold si ottiene dall'intersezione delle 4 curve in fig. 3.9. Per via delle fluttuazioni statistiche a cui sono soggetti i dati non è possibile stimare in maniera precisa il valore di threshold, ciononostante il valore di threshold atteso, ovvero $p \approx 10\%$, è compatibile con i dati ottenuti, in quanto per $p \approx 10\%$ le 4 curve sono sufficientemente vicine.

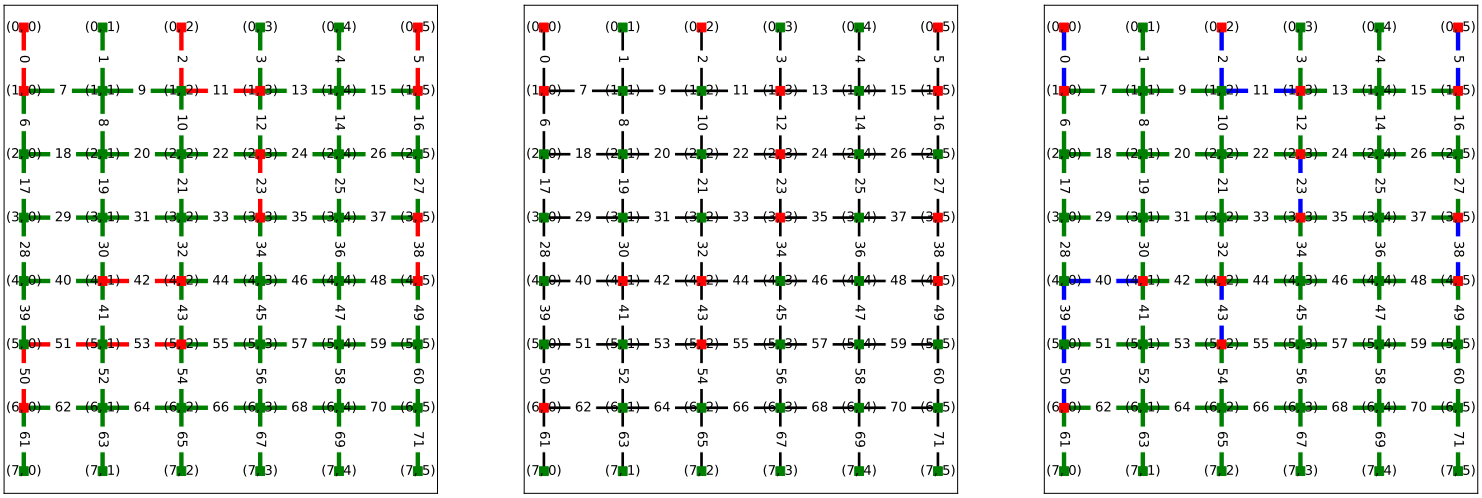

Figura 3.7: Probabilità di errore $p = 10\%$, numero di qubit totale $n = 72$. Nella fig. di sinistra sono rappresentati i qubit sui quali è avvenuto un errore. Al centro sono rappresentati gli autovalori corrispondenti: in rosso gli autovalori con valore -1, in verde quelli con autovalore $+1$. A destra è rappresentata la correzione calcolata dal decoder, dove gli spigoli blu indicano l'operatore di correzione. La correzione ha avuto esito positivo.

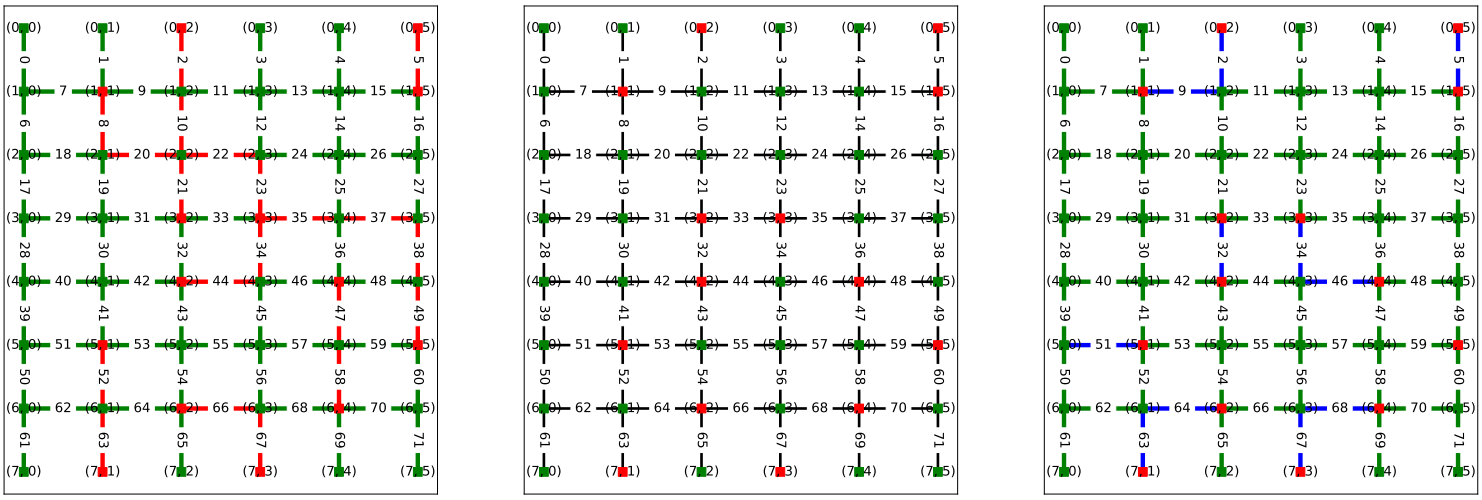

Figura 3.8: Generazione degli errori con probabilità di errore $p = 20\%$ e numero di qubit totale $n = 72$. Nella fig. di sinistra sono rappresentati i qubit sui quali è avvenuto un errore. Al centro sono rappresentati gli autovalori corrispondenti: in rosso gli autovalori con valore -1, in verde quelli con autovalore $+1$. A destra è rappresentata la correzione calcolata dal decoder, dove gli spigoli blu indicano l'operatore di correzione. La correzione ha avuto esito negativo.

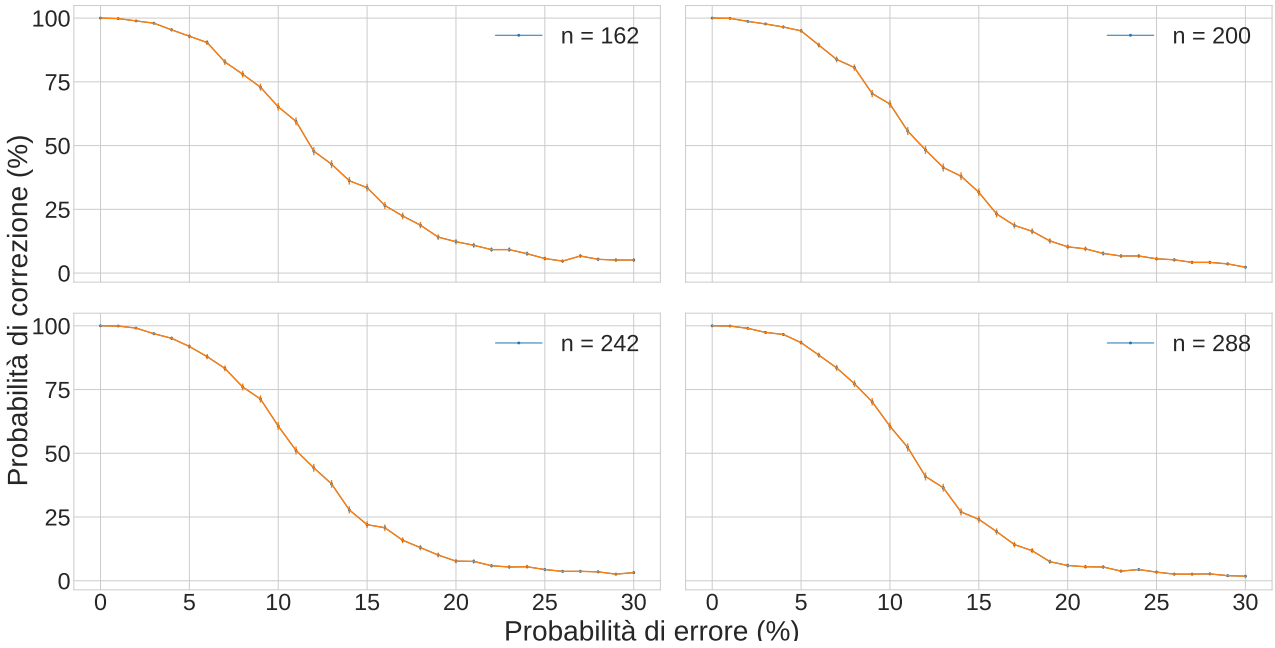

Figura 3.9: I grafici rappresentano la probabilità di correzione in funzione della probabilità di errore p. Per ogni punto vengono effettuate $N_s = 1000$ simulazioni di correzione. Le barre di errore sono presenti anche se non ben visibili. Le curve ottenute sono tra di loro compatibili, tuttavia presentano delle fluttuazioni statistiche dovute al basso numero di simulazioni di correzioni effettuate.

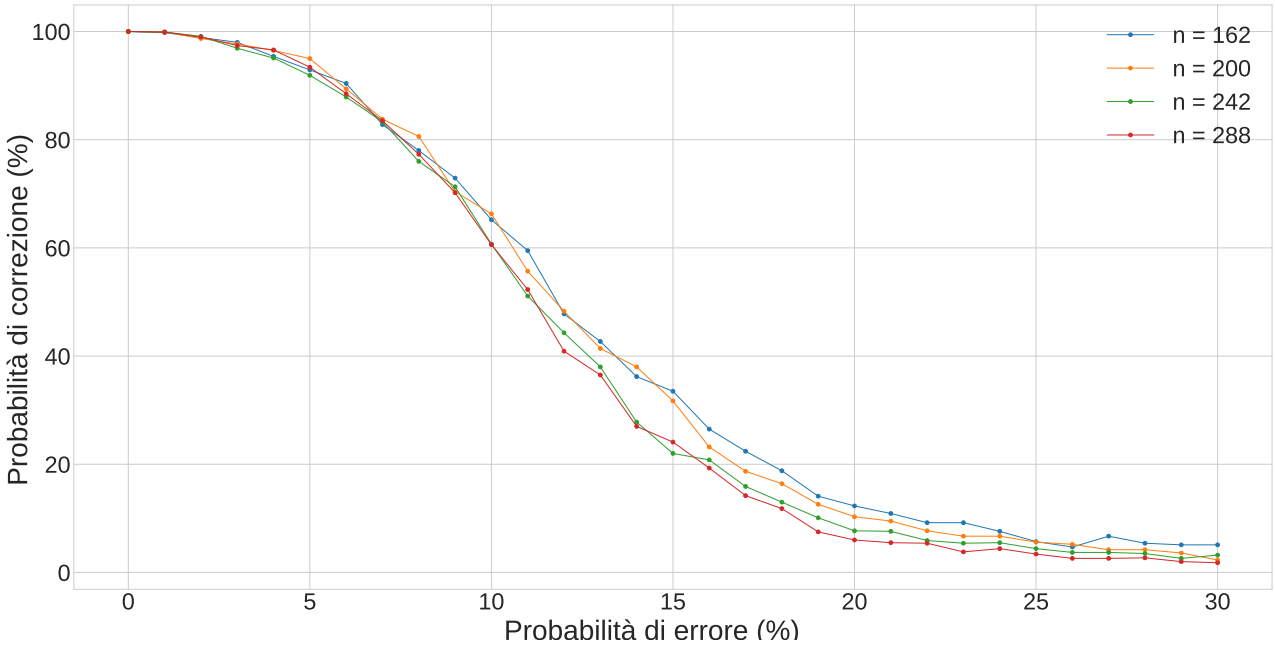

Figura 3.10: In figura è rappresentata la sovrapposizione delle 4 curve di fig. 3.9. Le barre di errore sono omesse. Non è possibile stimare con precisione l'errore di threshold, tuttavia notiamo che il valore atteso di threshold, $p \approx 10\%$, è compatibile con i dati ottenuti dalla simulazione.

Conclusione

In questa tesi viene trattato il tema attuale della *quantum error correction*. Lo studio di questa branca dell'informatica quantistica è essenziale per poter realizzare sistemi informatici quantistici affidabili. In particolare, dopo una breve introduzione sulla meccanica quantistica e sulla computazione quantistica, è stato descritto il problema della *quantum error correction* attraverso il formalismo degli stabilizer. Viene descritto il codice di correzione del toric code e del planar code. Viene, inoltre, proposta un'implementazione di decoder per il planar code basata sull'algoritmo di Edmond e scritta in python. I risultati della simulazione confermano il buon funzionamento dell'implementazione del decoder. L'accuratezza della simulazione è limitata dall'elevata potenza di calcolo necessaria per simulare un grande campione di dati, ciononostante i risultati ottenuti rispecchiano quelli attesi in base al modello teorico.

Bibliografia

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).

[2] D. Brown, "Topological codes and computation," (A lecture course given at the University of Innsbruck, 2014) pp. 1–46.

[3] R. Zucchini, "Quantum physics lecture notes," (Università di Bologna, 2021).

[4] D. Gottesman, *Stabilizer codes and quantum error correction*, Ph.D. thesis, California Institute of Technology (1997).

[5] A. Kitaev, "Fault-tolerant quantum computation by anyons," Annals of Physics **303**, 2 (2003).

[6] M. B. Elliott, "Stabilizer states and local realism," (2008), arXiv:0807.2876 [quant-ph] .

[7] R. Diestel, *Graph Theory* (Springer, 2017).

[8] Z. Galil, "Efficient algorithms for finding maximal matching in graphs." (1983) pp. 90–113.

[9] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," Journal of Mathematical Physics **43**, 4452 (2002), https://doi.org/10.1063/1.1499754 .

[10] "Matching (graph theory) - wikipedia," `https://en.wikipedia.org/wiki/Matching_(graph_theory)`.

[11] C. Cesare, *Topological Code Architectures for Quantum Computation*, Ph.D. thesis, University of New Mexico (2014).

[12] A. Hagberg, P. Swart, and D. Chult, "Exploring network structure, dynamics, and function using networkx," `https:`

//www.researchgate.net/publication/236407765_Exploring_Network_Structure_Dynamics_and_Function_Using_NetworkX (2008).