



M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem

TDK DOLGOZAT

Kvantumos hibajavítás áramköri szintű zajelemzése a felületi kódon

Készítette:

Domokos Bálint

Fizika BSc

BME Természettudományi kar

Konzulens:

Dr. Asbóth János

Egyetemi docens

BME Elméleti Fizika Tanszék

2022. november 1.

Kivonat

Összetett kvantumos algoritmusok futtatására alkalmas kvantumszámítógép építéséhez elengedhetetlen a kvantumos hibajavítás. A kvantumbitek rövid koherenciaideje mellett problémát jelentenek a gépek működése közben fellépő, pl. kalibrációs hibák is, ezek torzításai ellen aktívan védeni kell a kvantuminformációt. E célt szolgálják a hibajavító kódok, ahol egy-egy logikai kvantumbitet több fizikai kvantumbit összefonódott állapotában tárolunk, a környezeti zajoktól védett módon. Az összefonódott állapoton végzett megfelelő mérések segítségével információt nyerünk a kvantumos memória sérüléséről. A mérési eredmények ismeretében pedig megpróbáljuk kijavítani a hibát. Az egyik legígéretesebb kvantumos hibajavító konstrukció az úgynevezett felületi kód.

Dolgozatomban a $d=3$ kódtávolságú felületi kódot vizsgáltam, mint kvantumos memóriát. Ennek kvantumos áramkörü szintű kivitelezéséhez $9+8$ kvantumbitre van szükség. A környezet zaját a kvantumbiteken kétféleképpen modelleztem: véletlenszerűen alkalmazott Pauli-Z operátorokkal, illetve úgynevezett koherens-Z hibákkal. Numerikus szimulációval, a qiskit python szoftverfejlesztő készlettel meghatároztam a logikai kvantumbit hűségét a fizikai kvantumbitek környezeti és kiolvasási hibájának függvényében.

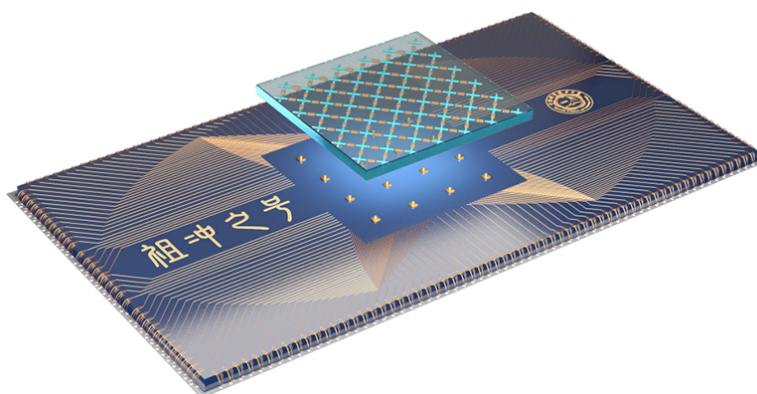
Tartalomjegyzék

1. Bevezetés	3
2. Kvantumos hibajavítás stabilizátorformalizmusban	5
2.1. Pauli-operátorok	5
2.2. Stabilizátorcsoport	6
2.3. Logikai operátorok	6
2.4. Környezeti zajok	7
2.5. Stabilizátormérés	7
2.6. Operátorok ekvivalenciája	8
2.7. Rendszer inicializálása a kódoló altérbe	8
2.8. Dekódolás és javítás	8
2.9. Stabilizátor kód menete	8
3. Felületi kód	9
3.1. $D=3$ kódtávolságú felületi kód	9
3.2. Felületi kód áramköri szintű kivitelezése	10
3.3. 3D dekódolás a felületi kódon	13
4. Vizsgált hibacsatornák	16
4.1. Z hibák tökéletes kvantumos kapuk mellett	16
4.1.1. Dinamikus szétcsatolás	17
4.2. Hibával terhelt kvantumos kapuk Z hiba mellett	17
5. 17 kvantumbites zajos felületi kód numerikus szimulációja	18
5.1. Trajektória generálása Qiskit-tel	18
5.2. Dekódolás és javítás	18
5.3. Szimuláció	19
5.4. Egy kvantumbit teljesítménye	20
6. Szimulációs eredmények	22
6.1. Z hibák tökéletesen kivitelezett kvantumos kapuk mellett	22
6.2. Z hibák hibával terhelt kvantumos kapuk mellett	23
7. Konklúzió	24

1. Bevezetés

Nagy kapacitású kvantumszámítógépek megjelenésével új utak nyílhatnak meg egyes klasszikusan túl nehéznek bizonyuló problémák megoldására. A kvantumszámítógépek elterjedése első körben biztosan nem jelenti a klasszikus számítógépek lecserélődését, hiszen azok jelenlegi elképzeléseink szerint specifikus problémák megoldására lesznek alkalmasak. Sok témában lehet hallani, hogy áttörést hozhat a kvantumszámítógép megjelenése. Az egyik legrégebben beazonosított ilyen terület a kriptográfia. A jelenlegi titkosítási eljárások nagy része (pl.: RSA 2048) azon alapul, hogy nagyon nagy számok faktorizációs problémájának megoldására nincs klasszikusan polinomiális idővel skálázódó algoritmus. A probléma kvantumos megoldása a Shor algoritmus, viszont majdnem 30 éve ismert [1].

A Shor algoritmussal egy 10^n nagyságrendű szám faktorizációjához $10n$ kvantumbit-re van szükségünk [2]. Ezekon a kvantumbiteken a számítás n^3 időlépéssel végezhető el. Az RSA 2048 feltöréséhez tehát nagyjából 6000 kvantumbitre lenne szükség 10^{-10} -es hibaráta mellett. A jelenlegi kvantumos hardverek ezektől a paraméterektől azonban bőven elmaradnak. A legújabb szupravezetésen alapuló kvantumos chip-eken még két nagyságrenddel kevesebb kvantumbit van, például a kínai Zuchongzi chipen 66 kvantumbit-et tudtak elhelyezni [3]. Az RSA feltöréséhez nemcsak a kvantumbitek száma nem elegendő, hanem a kapuműveletek pontossága sem megfelelő, és a kvantumbitek koherenciaideje is túl rövid.



1.0.1. ábra. 66 darab szupravezető kvantumbitből készült Zuchongzi kvantumprocesszor[3]

A kvantumos memória védelmére, a klasszikus ismétléses kód mintájára, megjelentek a kvantumos hibajavító kódok. A kvantumos hibajavító kódokban több kvantumbit összefonódott állapotába kódoljuk az információt, ezzel lehetőségünk nyílik hibák javításra. A hibajavításnak és az ezzel járó nagyobb pontosságnak azonban ára van: egy számításra használható logikai kvantumbit létrehozásához sok fizikai kvantumbit-et kell használnunk. A kvantumos hibajavító kódok közül jelenleg az úgynevezett felületi kód tűnik a legbiztosabb megoldásnak [4]. Ezért a Google kvantumszámítógép fejlesztő részlege is olyan elrendezésben helyezte el kvantumbitjeit a chipen, hogy az felületi kód alkalmazására legyen alkalmas. Tudományos kísérleteikben többek között a felületi kódot is vizsgálják [5].

Ebben a dolgozatban egy 17 fizikai kvantumbites felületi kódban tárolt logikai kvantumbit érzékenységét vizsgáljuk környezeti zajokra és kalibrációs hibákra. A 2. fejezetben bevezetjük a stabilizátorformalizmus alapfogalmait. A 3. fejezetben definiáljuk a 17 kvantumbites felületi kódot és a hozzá tartozó stabilizátorokat. A 4. fejezetben mutatjuk be azokat a folyamatokat, amelyek által okozott hibákat vizsgáljuk. Az 5. fejezetben a Qiskit-re épülő numerikus szimuláció módszerét ismertetjük. A numerikus szimulációval kapott eredményeket, a logikai kvantumbit hibájának a fizikai hibavalószínűségtől való függését a 6. fejezetben mutatjuk be. Végül a 7. fejezetben konkludálunk.

2. Kvantumos hibajavítás stabilizátor formalizmusban

Kvantumos hibajavító kódok állapotvektorokkal történő leírása a kvantumbitek számának növelésével gyorsan nehezen követhetővé válik, mivel a rendszert leíró Hilbert-tér dimenziója a kvantumbitek számával exponenciálisan nő. Ezért bizonyos hibajavító kódok vizsgálatához érdemes az úgynevezett stabilizátor formalizmust használnunk [6][7].

2.1. Pauli-operátorok

Amint a kvantuminformáció általános elméletében, a stabilizátor formalizmusban is kulcsszerepet játszanak a Pauli operátorok. A Pauli-operátorokat kiegészítve az identitással X , Y , Z , I -vel fogjuk jelölni.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (2.1.1a)$$

$$X = |1\rangle\langle 0| + |0\rangle\langle 1| = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (2.1.1b)$$

$$Y = i|1\rangle\langle 0| - i|0\rangle\langle 1| = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad (2.1.1c)$$

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (2.1.1d)$$

A Pauli-operátorok többek között: (i) unitér és önadjungált operátorok, (ii) kommutálnak vagy antikommutálnak egymással (pl.: $XZ = -ZX$), és (iii) bázist alkotnak a 2×2 -es mátrixok terén. Mindhárom Pauli-operátor kifejezhető a másik kettő szorzata segítségével, ezek közül számunkra az $Y = iXZ$ összefüggés lesz a legfontosabb. Az egykvantumbites Pauli-operátorok $\{I_i, X_i, Y_i, Z_i\}$ tenzorszorzataival generálhatóak az n -kvantumbites Pauli-operátorok. Például $n = 3$ esetén egy ilyen operátor: $X \otimes I \otimes Z$, a továbbiakban az egyszerűség érdekében a tenzorszorzás jelet elhagyjuk, így a jelölés: XIZ . Az n -kvantumbit Pauli operátorokra a tenzorszorzat tulajdonságai miatt szintén fennállnak az alábbi tulajdonságok: (i) unitér és önadjungált operátorok (± 1 sajátértékek), (ii) kommutálnak vagy antikommutálnak egymással, és (iii) A $2^n \times 2^n$ -s mátrixok terén bázist alkotnak. Kiegészíthetjük az operátorokat úgy, hogy mind a 4^n operátor felvehet egy ± 1 vagy $\pm i$ fázist, így csoportot kapunk. (Ekkor viszont az elemek négyzete ± 1 , illetve nem minden csoportelem lesz hermitikus). A későbbiekhez fontos definiálnunk egy n -kvantumbit operátor (Hamming-)súlyát, ami az a szám, ahány kvantumbiten az adott operátor hatása nem az identitás operátor.

A kvantumos információt a környezettől való védelem érdekében több kvantumbit összefonódott állapotába kódoljuk bele. Ilyen kód például a klasszikusan is használt ismétléses kód, vagy az általam vizsgált felületi kód. Ezeket hibajavító kódoknak nevezzük, mivel a struktúrájuk lehetőséget ad hibák feltérképezésére és legjobb tudásunk szerinti javításra. A hibajavító kódok egyik fajtája a stabilizátorkódok, amelyek leírhatók a stabilizátor formalizmus segítségével.

2.2. Stabilizátorcsoport

A stabilizátor formalizmusban használt operátorok két halmazra oszthatók: a stabilizátor csoportra, illetve a logikai operátorokra. Amint a név is sugallja, csoportelméleti szempontból, a stabilizátor csoport elemei azok az n -kvantumbit Pauli-operátorok (a rendszer n kvantumbitet tartalmaz), amelyek a kód $|\psi_L\rangle$ állapotára invariánsan hatnak (n -kvantumbit Pauli csoport részcsoportha):

$$S|\psi_L\rangle = |\psi_L\rangle \quad (2.2.1)$$

Ebből belátható, hogy a stabilizátor csoport elemei egymással kommutálnak:

$$S_1 S_2 |\psi_L\rangle = S_1 |\psi_L\rangle = |\psi_L\rangle = S_2 S_1 |\psi_L\rangle \quad (2.2.2)$$

A rendszer legegyszerűbb leírásához viszont nem szükséges az összes operátort használnunk, hanem elég a stabilizátor operátorok egy független generátorhalmaza. Egy generátor halmaz $\{s_i\}$ független, ha

$$\prod_i s_i^{a_i} = I, \quad (2.2.3)$$

akkor és csak akkor, ha minden $a_i = 0$.

2.3. Logikai operátorok

A rendszert kvantum memóriaként szeretnénk használni, ezért annak $|\psi_L\rangle$ logikai kvantumbitet reprezentáló állapota a teljes állapottér egy kétdimenziós alterébe esik. Ezt nevezzük kódoló altérnek. Mi most az egyszerűség kedvéért egyetlen logikai kvantumbit tárolását vizsgáljuk. Az altéren vett bázisállapotokat nevezzük logikai nullának $|\bar{0}\rangle$ és logikai egynek $|\bar{1}\rangle$. A logikai operátorok $\bar{X}$ és $\bar{Z}$ az alábbi módon hatnak a bázis állapotokra:

Logikai $\bar{X}$

$$\bar{X}|\bar{0}_L\rangle = |\bar{1}_L\rangle \quad (2.3.1a)$$

$$\bar{X}|\bar{1}_L\rangle = |\bar{0}_L\rangle \quad (2.3.1b)$$

Logikai $\bar{Z}$

$$\bar{Z}|\bar{0}_L\rangle = |\bar{0}_L\rangle \quad (2.3.2a)$$

$$\bar{Z}|\bar{1}_L\rangle = -|\bar{1}_L\rangle \quad (2.3.2b)$$

Látható, hogy a logikai operátorok is hasonló tulajdonságokkal bírnak, mint az 1 kvantumbit Pauli-operátoraink, tehát önmaguk inverzei és egymással antikommutálnak. A kvantum kódok egyik fontos jellemzője a távolságuk, ami definíció szerint az $\bar{X}$ és $\bar{Z}$ logikai operátorok súlyának a minimuma. A logikai operátorok a kódoló altéren belül tartják a kódot. Mivel a stabilizátor csoport elemei definíció szerint invariánsan hagyják a kód állapotát, a stabilizátorok és a logikai operátorok egymással felcserélnek, másrészt a logikai operátorok szorzata bármelyik stabilizátor elemmel egy olyan operátor, ami azonosan hat, mint az adott logikai operátor ($\bar{L}$):

$$\bar{L}S|\psi_L\rangle = \bar{L}|\psi_L\rangle \quad (2.3.3)$$

2.4. Környezeti zajok

A fizikai rendszeren nem tudjuk teljesen elszigetelni a külvilágtól, így a környezet hatást fejt ki az állapotra. A környezethez hasonlóan a működéshez szükséges beavatkozásaink sem tökéletesek, így például kalibrációs hibák léphetnek fel. Ezeket az effektusokat leírhatjuk különböző E hibaoperátorokkal:

$$E |\psi_L\rangle = |\tilde{\psi}\rangle \quad (2.4.1)$$

Azt, hogy milyen hibaoperátorokkal érdemes modelleznünk a rendszert, nagyban függ a hibajavító kód fizikai implementációjától. A legáltalánosabb hibacsatornák, amivel a zajt modellezni szoktuk a Pauli-hibacsatornák. Ezekben a hibajavító kód kvantumbitjein valamilyen meghatározott időegységenként p valószínűséggel történik hiba, mely hiba operátor egy Pauli operátor. Ilyen hibacsatorna például a független-zaj, melyben egymástól függetlenül p valószínűséggel történik egy Z- (fázis-hiba) vagy X-hiba (bitátfordulás). A rendszer tökéletlen szigetelését leírhatjuk úgynevezett koherens hibákkal, amik a rendszer ciklusidejével arányos unitér forgatások az egyes kvantumbiteken.

2.5. Stabilizátormérés

A kódoló altér állapotai a stabilizátor operátorok $+1$ sajátértékű sajátállapotai. Amennyiben a rendszer $|\psi_L\rangle$ állapota valamilyen hibahatás miatt kikerül a kódoló altérből, lesz olyan stabilizátor operátor, melynek -1 sajátértékhez tartozó sajátállapotába kerülünk. (A stabilizátor operátorok n -kvantumbit operátorok, vagyis ± 1 sajátértékük két sajátaltérre osztja a Hilbert-teret). Ezért két csoportra oszthatóak a hibák, amelyek a rendszer állapotát akaratunkon kívül manipulálják: vannak a kódoló altérből kivezető és a kódoló altérben tartó hibák. A stabilizátor operátorok mérhetőek, ezért azok mérése az említettek alapján alkalmas a kódoló altérből kiléptető hibák detektálására. Írjuk le az E hibát egy n -kvantumbit Pauli-operátorral. Az E operátor, vagy kommutál vagy antikommutál a stabilizátor elemeivel. Amennyiben kommutál,

$$ES |\psi_L\rangle = SE |\psi_L\rangle = E |\psi_L\rangle, \quad (2.5.1)$$

akkor a hiba a kódoló altérben marad, vagyis a stabilizátor mérések $+1$ sajátértékkel térnek vissza. Ilyenkor E -t nem detektálható hibának nevezzük. Valamelyik stabilizátor elemmel való antikommutáció esetében viszont

$$ES_i |\psi_L\rangle = -S_i E |\psi_L\rangle = -E |\psi_L\rangle, \quad (2.5.2)$$

tehát $E S_i$ stabilizátor -1 -es alterébe vitte az állapotot, ezért ezt méréssel észrevesszük, és így E egy detektálható hiba.

A stabilizátor elemeinek sajátaltereibe vetítő projektorok az alábbi formában írhatóak fel:

$$P_{\pm} = \frac{I \pm S}{2}. \quad (2.5.3)$$

Ezekkel a projekciókkal projektív méréseket konstruálhatunk. Ugyanakkor egyértelmű, hogy felesleges minden stabilizátor elemet mindig végigmérni, elegendő csupán egy független generátor halmazukat megmérni az azonos mennyiségű információ begyűjtéséhez. Egy független generátor halmaz mérési eredményét nevezzük szindrómának.

2.6. Operátorok ekvivalenciája

Ahogy azt már korábban láttuk, egy logikai operátor stabilizátor elemmel megszorozva, ugyanolyan hatással van a kódra, mint az eredeti logikai operátor. Hasonlóan a hibáknál egy E hibaoperátor és egy stabilizátor szorzata ugyan oda juttatja a kódot. Érdekes ezért az operátoraink között egy ekvivalencia relációt bevezessünk. Két operátor a stabilizátor formalizmusban ekvivalens, amennyiben azok egymástól stabilizátorgenerátor elemek szorzatában térnek el.

$$O_1 \sim O_2, \text{ ha} \\ \exists \{a_j\}_{j \in J} : O_1 = \left(\prod_j s_j^{a_j} \right) O_2. \quad (2.6.1a)$$

2.7. Rendszer inicializálása a kódoló altérbe

A kódoló altér definíció szerint a Hilbert-tér azon altere, melynél minden stabilizátor a $+1$ sajátállapotában van. A rendszert egy $|\Psi\rangle$ kiinduló állapotból a stabilizátor mérésnél már ismertetett $+$ -os projektorok segítségével tudjuk a kódoló altérbe juttatni.

$$\prod_i \frac{I + s_i}{2} |\psi\rangle = |\psi_L\rangle \quad (2.7.1)$$

2.8. Dekódolás és javítás

A szindróma által szolgáltatott információ alapján különböző hipotéziseket állíthatunk fel a hibára. Egy meghatározott elv szerint a kiválasztunk egyet hibahipotézisek közül, amiről azt gondoljuk, hogy alkalmas javítani az esetleges hibát. Ezt a folyamatot nevezzük dekódolásnak. A dekódolás után egy hibajavító operátort alkalmazva próbáljuk visszajuttatni a kódot az eredeti állapotba. A hibajavító operátorok a rendszer állapotát mindig a kódoló altérbe viszik.

2.9. Stabilizátor kód menete

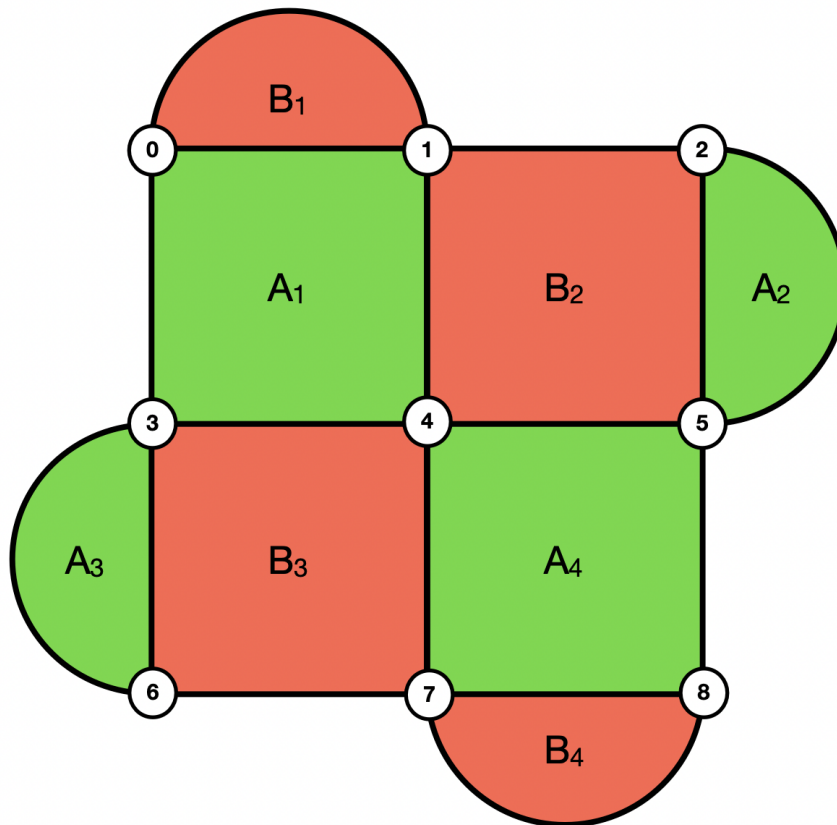
Összegezve, a stabilizátor kódokban a következő protokoll szerint tudunk védekezni a környezeti zaj hatásával szemben:

1. Végigmérjük a stabilizátor generátor halmazt, ezt nevezzük egy ciklusnak.
2. A ciklusban kapott mérés eredménye a szindróma, amelyet a dekódolás folyamán értelmezünk.
3. A dekódolás eredménye alapján javítunk.

3. Felületi kód

A legígéretesebbnek tűnő kvantumos hibajavító kód konstrukció a felületi kód[4]. A felületi kódban négyzetrácsos elrendezésben helyezzük el az adattároló kvantumbiteket ("data quantum bit"). A legkisebb ilyen elrendezés, melyben logikai kvantumbitet tudunk tárolni, a $d=3$ szélességű rács. A felületi kód működését ezen a példán keresztül mutatom be.

3.1. $D=3$ kódtávolságú felületi kód



3.1.1. ábra. A $d=3$ kódtávolságú felületi kódban háromszor hármass rácspan elhelyezett kvantumbitek (számozott fehér körök). A kód stabilizátor generátorai a lapszomszédos kvantumbitek együttes X vagy Z paritását adják. Az X stabilizátorokhoz a zöld lapok mentén elhelyezkedő kvantumbitek tartoznak, míg a Z stabilizátorokhoz a piros lapok mentén elhelyezkedők.

A 3.1.1. ábrán látható a $d=3$ kvantumbit rácsszélességű felületi kód elrendezése. A kód a stabilizátorkódok csoportjába tartozik, a stabilizátorgenerátorai a térben szomszédos kvantumbitek együttes X és Z paritásoperátorai, tehát ez alapján megkülönböztetünk Z stabilizátorokat (Plakett-operátorok, B_i) és X stabilizátorokat (Vertex operátorok, A_i).

X-stabilizátor	Z-stabilizátor
$A_1: X_0X_1X_3X_4$	$B_1: Z_0Z_1$
$A_2: X_2X_5$	$B_2: Z_1Z_2Z_4Z_5$
$A_3: X_3X_6$	$B_3: Z_3Z_4Z_6Z_7$
$A_4: X_4X_5X_7X_8$	$B_4: Z_7Z_8$

1. táblázat. A $d=3$ kódtávolságú felületi kód stabilizátor csoportjának generátorai

A 1. táblázatban látható a $d=3$ rácsszélességű felületi kód stabilizátorcsoportjának független generátorhalmaza. Megfigyelhető, hogy egy X és egy Z stabilizátornak vagy nulla vagy kettő közös fizikai kvantumbitje van, így azok kommutálni fognak. A logikai operátorok bemutatása előtt érdemes megvizsgálni X és Z hibák hatását a felületi kódra. Pauli-operátorok segítségével sok hibacsatorna leírható, így egy intuitív kép kialakításához elegendő lehet ezek hatását megvizsgálni a felületi kódra. Az X operátor z -bázisban egy kvantumbiten megfordító hatású ("bit flip"), tehát amennyiben egy Z stabilizátorhoz tartozó kvantumbitek közül páratlan számú történik X hiba, akkor a stabilizátor értéke megváltozik. Hasonlóan a Z operátor pedig x -bázisban megfordító hatású, így egy X stabilizátoron páratlan Z hiba okoz paritás változást.

A kódra ható logikai operátorok nem visznek ki a kódoló altérből, tehát egyik stabilizátor $+1$ paritását sem változtatják meg, viszont mégsem ekvivalens operátorok az identitással. Ilyen operátorok a képen látható elrendezésben a rács tetejét és alját összekötő Z sztringek, illetve a két oldalt összekötő X sztringek. Például

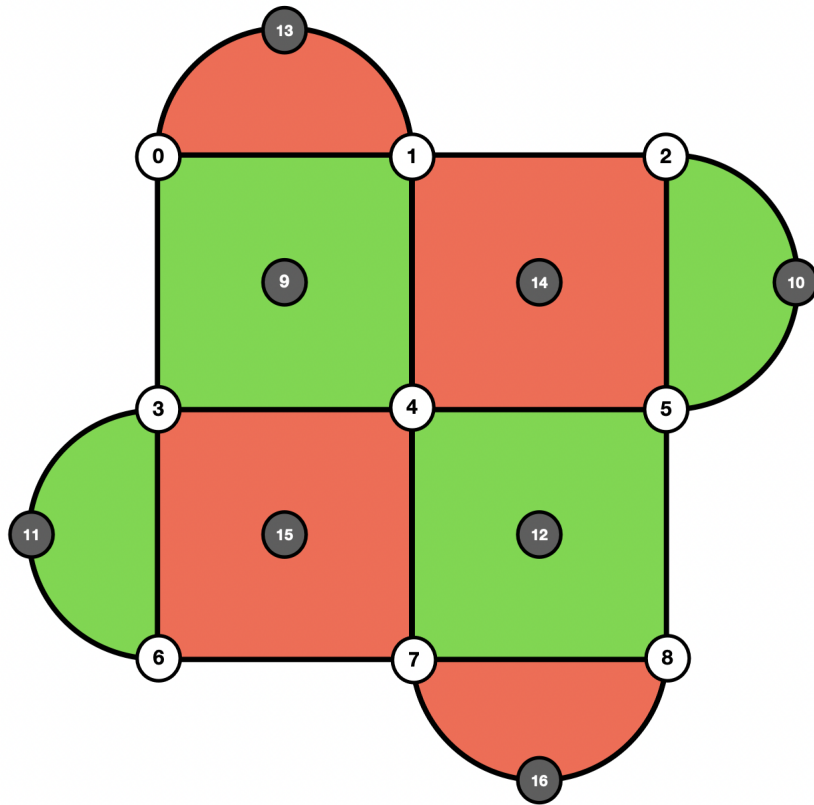
$$\bar{X} = X_0X_1X_2, \quad (3.1.1a)$$

$$\bar{Z} = Z_0Z_3Z_6. \quad (3.1.1b)$$

Az elrendezés tetején a jobb oldalról hiányzik egy vertex operátor (zöld), ezt nevezzük durva határnak. A bal oldal tetejéről és jobb oldal aljáról hiányzik egy plakett operátor (piros), az ilyen pedig sima határnak hívjuk. Általános esetben a logikai operátorokat a határok segítségével azonosíthatjuk be. A logikai $\bar{X}$ operátor a durva határokat, míg a logikai $\bar{Z}$ a sima határokat köti össze. A két ilyen sztring páratlan sokszor metszheti egymást. A metszéspontokban lévő kvantumbitekben a Z és X operátorok antikommutálnak egymással, tehát a két logikai operátor is antikommutálni fog az elvárásnak megfelelően. A logikai operátor tetszőleges megválasztása, az előírt feltételek mellett, azért lehetséges, mert a rács tetejét és alját összekötő bármelyik két Z sztring operátor ekvivalens, vagyis stabilizátor generátorokkal megszorozva egymásba vihetők. (Hasonlóan ez igaz az X logikai operátorokra is). Ebből adódóan a minimális súlyú logikai operátorok éppen a rácsszélességgel egyeznek meg. Tehát a most felvázolt kód $d=3$ -as kódtávolságú.

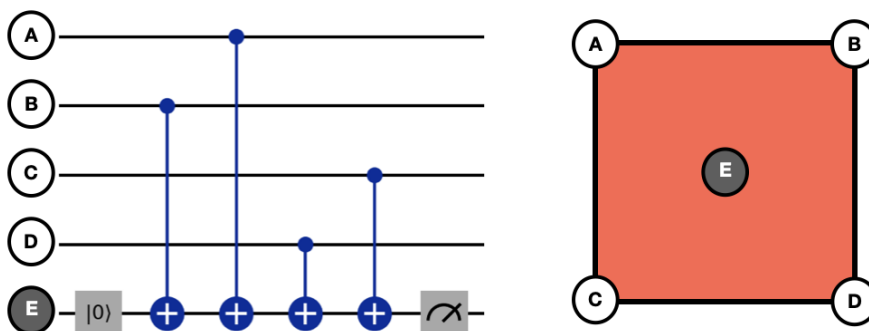
3.2. Felületi kód áramköri szintű kivitelezése

A kvantumos információ kódolásához és annak mérésekkel történő védelméhez további kvantumbitekre van szükség. A felületi kód ciklusokban rendre mind a nyolc stabilizátort meg szeretnénk mérni, méghozzá párhuzamosítva, így ehhez még további nyolc kvantumbitre van szükségünk. Az elrendezésbe az ancillákat behelyezve a 17 kvantumbites felületi kódot kapjuk:



3.2.1. ábra. A $d = 3$ kódtávolságú 17 kvantumbites felületi kód. A kilenc adattároló kvantumbit fehérrel a négyzetrács csúcsain helyezkedik el, A nyolc stabilizátorhoz tartozó ancillák, pedig szürke színezéssel szerepelnek.

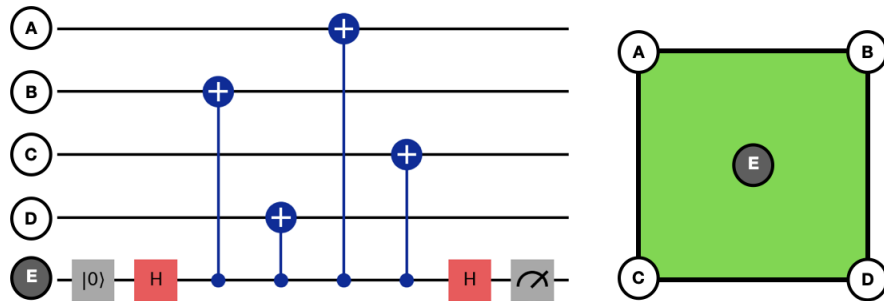
A stabilizátorokon 2.5.3-nak megfelelő projektív méréseket szeretnénk végezni. A Z-stabilizátorok esetében az együttes paritásmérés megvalósítható CNOT kétbites kapukkal a következőképpen: a stabilizátorhoz tartozó adattároló kvantumbiteket mint kontroll, a kezdetben $|0\rangle$ -ra inicializált ancillát pedig, mint target kötjük be, majd a CNOT-ok után megmérjük azt.



3.2.2. ábra. Z stabilizátor mérés kvantum áramköri rajza

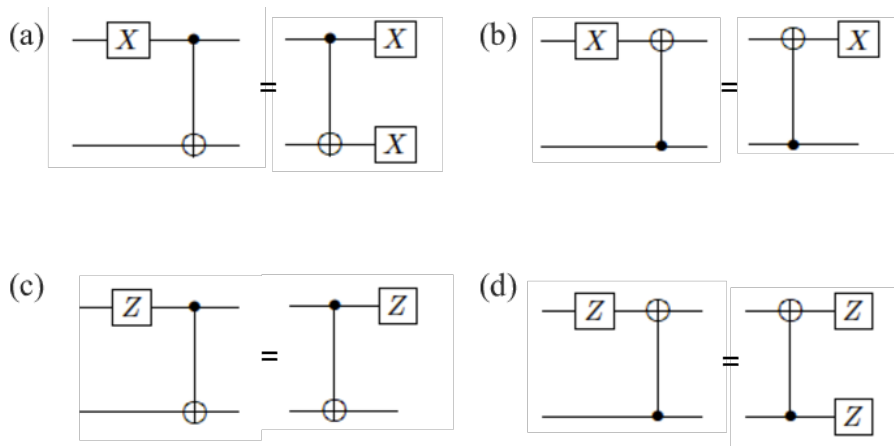
Az X-stabilizátorok a lapszomszédos kvantumbitek X paritását adják, ezek mérése nehezebb feladat. Az x-bázisban történő mérést megvalósíthatjuk úgy, hogy az adattároló kvantumbiteken a CNOT előtt és után egy Hadamard kaput alkalmazunk. Ennél viszont kevesebb kapuval is elvégezhető egy kapuazonosság segítségével. Az ancillát, megint

inicializáljuk $|0\rangle$ -ba és ezen rögtön végrehajtunk egy Hadamardot. A Hadamard után jönnek a CNOT-ok, viszont ezeket most az előzőhöz képest fordítva kell bekötni, tehát az ancilla szerepel kontrollként és az adattároló kvantumbitek targetként, végül az ancillán újra hat egy Hadamard kapu és megmérjük azt.



3.2.3. ábra. X stabilizátor mérés kvantum áramköri rajza

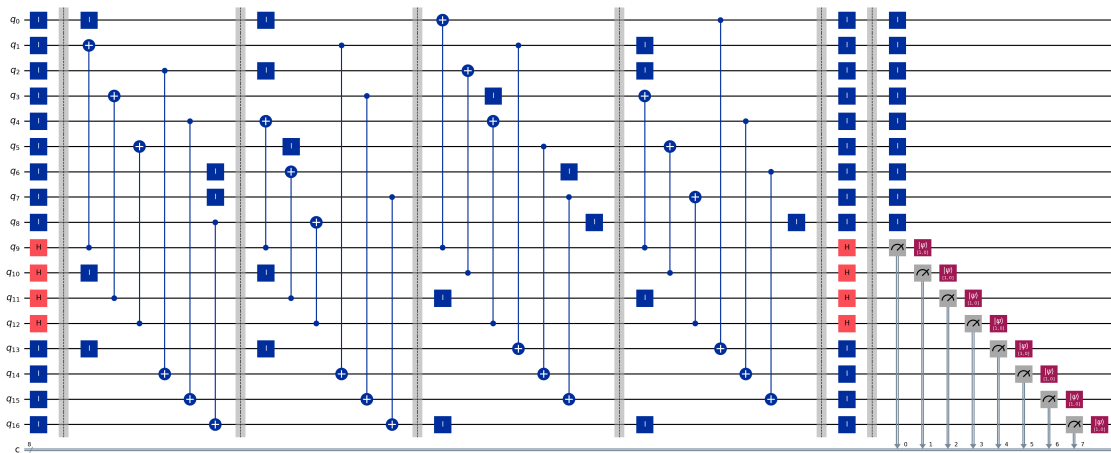
A mérés időtartamának minimalizálásához szeretnénk a műveleteket párhuzamosítani. Ekkor viszont figyelniük kell a CNOT-ok bekötésének sorrendjére [8].



3.2.4. ábra. Pauli hibák továbbterjedése CNOT-kapukon. Terjedés szempontjából egy Z , hiba veszélyes, ha az a target kvantumbiten történik. Egy X hiba a CNOT-on keresztül akkor terjed tovább, ha az előtte a kontroll kvantumbiten volt. Ábra: [9]

Amennyiben az egyes stabilizátorokat az óramutató járásával egyezően, vagy annak járásával ellentétesen (ABDC, vagy ACDB sorrendben) kötjük be, akkor a mérések nem fognak kommutálni. A ciklus során lesz olyan adattároló kvantumbit, ami először targetként, majd kontrollként szerepel. Így egy esetleges hiba továbbterjedhet. Ezt elkerülhetjük, amennyiben S vagy Z alakban kötjük be a CNOT-okat. A bekötési sorrend meghatározásánál még figyelembe kell veyük a logikai operátort megvalósító sztring irányára is. Az S vagy Z alakot úgy kell felvegyük, hogy merőleges legyen az a logikai operátor sztringjének irányára. Amikor összekötjük a kvantumbitek a CNOT bekötés sorrendje szerint, az akkor lesz merőleges, ha két összekötött él merőleges az ahhoz a stabilizátorhoz tartozó logikai operátorra. Például egy Z stabilizátor mérésnél, ha a második és a harmadik CNOT között egy Z hiba van az ancillán, az továbbterjed a második kettő adattároló kvantumbitre (C és D). Viszont ennek a hibasztringnek az irányára

merőleges a logikai Z operátorra, ezért ez a terjedés nem annyira veszélyes a logikai kvantumbitre. Vannak két kvantumbites stabilizátorok is, ezek esetében a bekötési sorrendet úgy tudjuk meghatározni, hogy képzeletben kiegészítjük a rácsot és a szokásos bejárás mentén haladunk, a virtuális kvantumbitekhez érve pedig nem csinálunk semmit. Így a CNOT-okat párhuzamosan négy lépésben végezzük el, rendre hat CNOT párhuzamos bekötésével. Ezért két ancilla és három adattároló kvantumbit ebben a szakaszban pihen, vagyis névlegesen egy I kaput hajtunk végre rajtuk.



3.2.5. ábra. A 17 kvantumbites felületi kód párhuzamosított stabilizátor mérésének kvantum áramköri rajza.

A teljes párhuzamosított kapcsolási rajz 3.2.5. ábrán látható, az egy időben végzett lépéseket a szürke vonalak határolják. A ciklus összesen $6+1$ részre tagolódik, az első hatban egy- és kétkvantumbites kapuműveleteket hajtunk végre, majd megmérjük az ancillákat és visszaállítjuk őket kiindulási állapotukba.

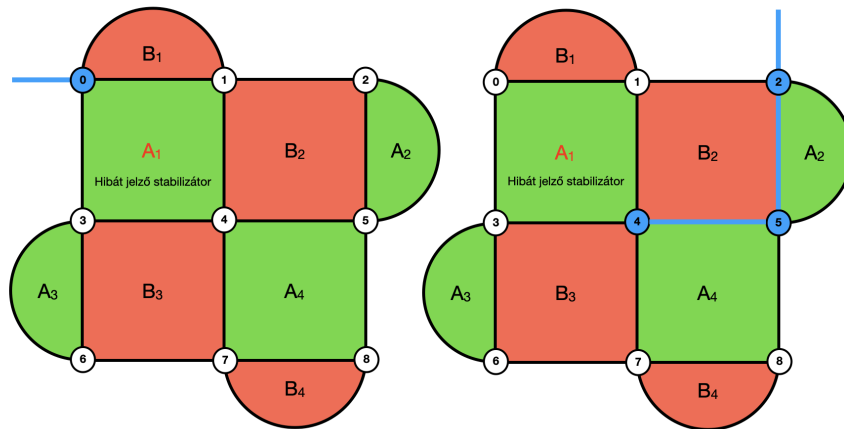
A tárolt információkra nem csak a kvantumbitekén történő fizikai hibák veszélyesek, hanem az is, hogy méréseket sem tudjuk tökéletesen elvégezni. Ez a mérési eredmény, vagyis szindróma dekódolási problémáját jelentősen megnehezíti. Egy kör felületi kód után nem tudhatjuk, hogy a stabilizátor esetleges jelzése fizikai hiba vagy kiolvasási (esetleg azzal ekvivalens inicializálási) hibának köszönhető. Emiatt nem végzünk javítást minden egyes mérés után, hanem több információt gyűjtünk, jellemzően a kódtávolságnak megfelelő ciklust futtatunk és a szindrómákat mentjük. Ahhoz, hogy a végállapot biztosan a kódoló altérbe kerüljön a legvégén, szimulációkban elvárjuk, hogy az utolsó mérési körben ne történjen hiba. A dekódolást pedig ezzel a 3 dimenziós (tér 2 dimenziója és idő) adathalmazon végezzük el.

3.3. 3D dekódolás a felületi kódon

A dekódolási probléma megoldására több eljárás is alkalmazható. Az egyik ilyen lehetőség a minimális súlyú tökéletes párosítás ("minimum weight perfect matching"), amit mi is használtunk a szimulációkhoz.

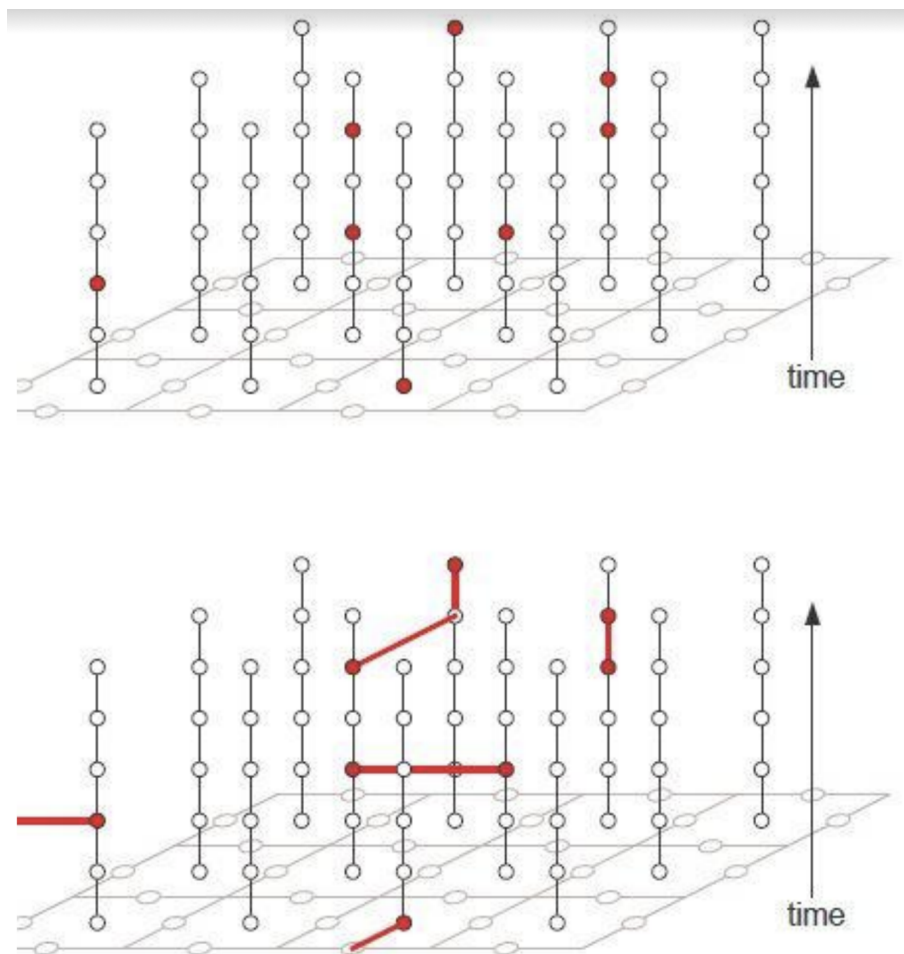
A három dimenziós eset előtt bemutatnám a két dimenziós esetet, melyben nem történhetnek hibás mérések, illetve csupán egy szindróma van. Ekkor úgy dekódolunk, hogy megkeressük azt a legkisebb súlyú Pauli operátort, ami az adott hibát okozhatta

és ezt újrarahatva javítjuk a hibát. A korrigáló operátor ezért olyan, hogy $+1$ paritású sajátállapotába kerül vissza az összes stabilizátor.



3.3.1. ábra. Az A_1 vertex operátor jelzett számunkra hibát. Ezt a szindrómát sokféle hiba okozhatta, kaphattuk egy a 0-ás kvantumbiten történő Z hibával (bal ábra). Keletkezhetett máshogy is például egy $Z_2Z_5Z_4$ sztring által is. Ekkor a minimális súlyú tökéletes párosítás értelemben a legkisebb súlyú hibahipotézist választjuk ki, tehát ebben az esetben a bal oldalt (egynél kisebb hibasúly nem lehetséges).

A három dimenziós esetben több zajos ciklus szindróma mérés eredményével dolgozunk. A szindrómákból első ránézésre nehéz következtetni a megelőző eseményekre. Ezért a mérési eredményekből legyártunk egy úgynevezett megfordulás mátrixot. A megfordulás mátrixban azokon a pozíciókon található 1-es, ahol a stabilizátor mérés előjelet váltott, és 0-ás ott, ahol a korábbi és a későbbi mérés egyezik. A szindróma mátrix információját tehát egy jobban áttekinthető formába alakítjuk.



3.3.2. ábra. Kiolvasási hibák esetén több ciklus mérési eredménye alapján végezzük a dekódolást. A stabilizátor mérések eredményeinek megváltozását rögzítjük (felső ábrán piros pöttyök jelölik a megváltozást). Majd a megváltozások alapján állítunk fel hibahipotéziseket, melyek közül a legkisebb súlyút kiválasztjuk. Az alsó ábrán egy ilyen hipotézis látható, ahol az egyes élek felelnek meg a hibáknak. A vertikális élek kiolvasási hibáknak, míg a horizontálisak, pedig fizikai hibáknak. Ábra forrása: Barbara Terhal: Surface Code Architecture, előadás

Például, ha a felületi kódban több ciklus alatt egy X hiba jelentkezett, ami egy Z stabilizátor értékét megváltoztatta, akkor ez egy olyan megfordulás mátrixot eredményez, amiben a megfelelő Z stabilizátornál egy 1-es van a hibás ciklus után és minden más helyen nullák. Egy hibás mérési eredménye ugyanakkor úgy jelentkezik, hogy az adott stabilizátornál két 1-es van egymás után, hiszen egyszer fordult a kiolvasási hiba miatt, aztán a következő ciklusban a helyes kiolvasás még egy fordulást indukált.

A feladat ebben az esetben is az, hogy megkeressük a legkisebb súlyú Pauli hibát mérési hibákkal keveredve, ami az adott szindróma sorozatot okozhatta. Egy lehetséges hibaszcenárió, úgy kaphatunk meg, hogy a megfordulásmátrix 1-es elemeit képzeletben „eltüntetjük”. A megfordulásokkal három dolgot végezhetünk: kiköthetjük őket oldalra, vagy összeköthetjük őket másik megfordulásokkal vertikálisan, illetve horizontálisan. A vertikális és horizontális összekötéseket használhatjuk vegyesen is. A horizontális összekötése két stabilizátor mérési megfordulásának azt jelenti, hogy a köztük lévő kvantumbit sztringen a stabilizátornak megfelelő hibák történtek. Két szindróma vertikális

összekötése kiolvasási hibát feltételez. A beköthető éleket a hibacsatornának megfelelően súlyokkal kell ellásuk. Ezeket a kiolvasási és adattároló kvantumbiteken történő fizikai hibák valószínűségének függvényében tudjuk meghatározni. Az egyes hibaszcenáriókat, így egy súlyfaktoral tudjuk ellátni. A minimális súlyú párosítási probléma pedig egy ilyen súlyozott gráfban keresi a legkisebb súlyú összekötést. A minimális súlyú párosítás hatékonyan megtalálható, erre az egyik bevett eljárás Edmond Blossom algoritmus[10]. A probléma megoldásának megfelelően meg tudjuk határozni a korrigáló operátorunkat.

4. Vizsgált hibacsatornák

A kutatás egyik célja a koherens és inkoherens hibák (Pauli-hibák) veszélyének összehasonlítása volt a felületi kódon. Ehhez külön csak koherens Z és csak Pauli Z hibákon alapuló aszimmetrikus hibacsatornákat alkalmaztunk, két konstrukció szerint.

4.1. Z hibák tökéletes kvantumos kapuk mellett

A 3.2.5 kvantumos áramkör egyértelműen széttagolható 7 egységre a szürke elválasztók alapján, amelyek az egy időben történő műveleteket tartalmazzák. Ezekhez az egységekhez hozzárendeltünk egy névleges időtartamot a jelenleg elérhető kvantumos hardverek paramétereinek megfelelően [5]. Az egy és két kvantumbites kapuk idejét 25 ns-nak vettük, az ancillák z -bázisban történő mérési idejét pedig 150 ns-nak. A kiolvasás ideje eltér a kísérletben jellemző értéktől. A rövidebb időtartam választásának oka, hogy nagyobb hangsúlyt kapjanak a cikluson belüli hibák. Az egy és két kvantumbites kapukat hibátlannak vettük, szimulált hardverünkben csak mérési hibák vannak jelen. Az egyes kvantumbiteken azokban az intervallumokban történhet hiba, amikor „pihennek”, tehát identitással hatunk. Két esetet vizsgáltunk: amikor (i) a hiba Z Pauli operátor, vagy (ii) egy z tengely körüli $\varphi \cdot \Delta t$ szöggel való forgatás ($E = e^{i\frac{\varphi\Delta t}{2}Z}$). A Pauli hibák valószínűségét úgy állítottuk be az inkoherens esetben, hogy a hiba bekövetkezésének valószínűsége egy pihenő szakaszban, egy r hibaráta és a névleges idő szorzata ($p(\Delta t) = r \cdot \Delta t$) legyen. Bevezettük a $p_{ch} = r \cdot T_{ciklus}$ mennyiséget, ami az egy felületi kód ciklus alatti hiba valószínűsége ($T_{ciklus} = 300$ ns). A kiolvasási hiba valószínűségét ehhez kapcsoltuk, a p_{ch} mennyiség a felének vettük. A koherens hiba esetében a φ értékét a következő egyenlet,

$$\sin^2\left(\frac{\varphi\Delta t}{2}\right) = r\Delta t, \quad (4.1.1)$$

alapján határoztuk meg. Amennyiben a koherens és az inkoherens hibák rátája között ez az egyenlőség áll fent, akkor egy kvantumbiten, amit időnként kiolvasunk a két hibacsatorna hatása azonos. Az X operátor $|+\rangle$ sajátállapotában inicializált kvantumbitet, ha Δt -vel később megmérünk x -bázisban, akkor ugyanazok a végkimeneteli valószínűségek rendelkezhetők mindkét hibacsatorna esetében.

Kódtávolság számú, jelen esetben $d = 3$ ciklust futtattunk. Az első két ciklusban az imént vázolt hibacsatornák szerint, a harmadik körben ugyanakkor csak a legelső lépésben (az X stabilizátor ancilláinak Hadamard kapui alatt) engedtünk meg hibát, utána már minden kapu és mérés tökéletes. Az utolsó körben a projektív értékű mérések vetítését azért nem rontjuk el, hogy a végállapot értelmezhető legyen.

4.1.1. Dinamikus szétcsatolás

Mivel a koherens hibák determinisztikusan rontják el a kvantumbitek állapotát, ezért a hatásukat eliminálni tudjuk egy trükkal a leghosszabb „pihenő” fázis közben. A mérés kezdete és az első kvantumbit kapuk befejeződése között összesen 175 ns telik el. Ebben az intervallumban van időnk több egy kvantumbites kaput is hajtatni. Koherens Z hibák hatását el tudjuk tüntetni, ha a pihenési időket megfelezzük és középidőben és az intervallum végén egy X kapuval hatunk, az alábbi egyenlőség miatt:

$$X e^{i \frac{\varphi \Delta t}{2} Z} X e^{i \frac{\varphi \Delta t}{2} Z} = I \quad (4.1.2)$$

Az X kapuk műveleti ideje miatt figyelni kell, hogy mikorra tesszük be őket, maximális hatáshoz, a két köztes időnek meg kell egyeznie. Ezt a módszert nevezzük dinamikus szétcsatolásnak. A vázolt hibacsatorna esetében megvizsgáltuk dinamikus szétcsatolással és anélkül is a koherens hibák hatását.

4.2. Hibával terhelt kvantumos kapuk Z hiba mellett

Megvizsgáltunk egy másik hibacsatorna konstrukciót is. Ebben a kvantumos műveletek (kapuk és mérés) kivitelezését pillanatszerűen végbemenőnek tételeztük fel, viszont azok hajtása között eltelt időt megtartottuk. Az előző esethez képest ez olyan változást okoz, hogy az egyes fázisokban azokon a kvantumbiteken is történhet hiba, amelyek abban az intervallumban valamilyen műveletben részt vettek. Ez felfogható úgy is, hogy a kapuk nem tökéletesek és nem feltétlenül a kívánt műveletet végzik el. Ezért ezt a konstrukciót nevezzük hibával terhelt kvantumos kapunak. Dinamikus szétcsatolást a koherens esetben viszont csak az utolsó és első fázis között hajtunk végre. A koherens és inkoherens rátákat most is egymáshoz igazítottuk a [4.1.1](#) szerint.

5. 17 kvantumbites zajos felületi kód numerikus szimulációja

Numerikus szimuláció segítségével vizsgáltuk a felületi kód ellenállóképességét környezeti zajra. A szimulációt python környezetben írtuk meg. A python kézenfekvő választás, mivel több szoftvercsomag is létezik hozzá, amelyek jól alkalmazhatóak kvantumos hibajavító kódok programozásához.

5.1. Trajektória generálása Qiskit-tel

Kvantumos áramkörök szimulációjához az IBM által fejlesztett Qiskit szoftverfejlesztő csomagot használtam. A Qiskitben összeállíthatunk a rendelkezésünkre álló kapukból egy tetszőleges kvantumos áramkört. A kvantumos áramkörök futtathatók többféle szimulátor segítségével, ezek közül mi az állapotvektor szimulátort választottuk. Az időfejlődés trajektóriákhoz elengedhetetlen volt számunkra, hogy a rendszer állapotát pontosan nyomon kövessük. Főlöszleges lett volna a nagyobb memória igényű sűrűségmátrix szimulációhoz folyamodni, mivel az általunk vizsgált csatornában végig tiszta állapotban maradt a rendszer.

A Qiskit segítségével összeállítottam a 3.2.5-en szereplő áramkör. Ehhez a 17 kvantumbit mellé 8 klasszikus bitet is alkalmazni kell, amelyekre a mérési eredményeket írjuk ki. Mindegyik kvantumbitét $|0\rangle$ -ba inicializáltuk. Ha egyszer hibátlanul futtatjuk az áramkört és javítunk (minimális súlyú javító operátorral), akkor legenerálhatjuk a $|\bar{0}\rangle$ -t. Ebből egy $\bar{X}$ operátor alkalmazásával kapjuk meg $|\bar{1}\rangle$ -et. Fontos, hogy az ancillákat a ciklus végén mindig nullába inicializáljuk. Ekkor az összefonódott rendszer szorzat állapotba kerül és megkaphatjuk külön az adattároló kvantumbitekhez tartozó hullámfüggvényt:

$$|\psi\rangle_{S17} = |0\rangle_{ancilla} \otimes |\bar{\psi}\rangle \quad (5.1.1)$$

Az áramkör inicializálását is úgy tehetjük meg, hogy elkészítjük egy tetszőleges logikai állapot vektorát, majd abból könnyen kiszámítható a nulla ancilla beállítási állapota a teljes rendszernek és azt betáplálhatjuk a Qiskit áramkörbe.

Az áramkört a három ciklusra nem tudjuk előre felépíteni, hiszen abban az inkohereus hibacsatorna esetén véletlenszerűen Z hatások kerülhetnek. Ezért, kiegészítő függvények segítségével, lépésenként adjuk hozzá az áramkörhöz az elemeket és szakaszonként futtatjuk azt. Az azonos idejű műveleteket megadó szakasz felépítésénél az inkohereus esetekben a numpy könyvtár segítségével generáltuk a véletlen számot, ami meghatározta, hogy I vagy Z kaput kell az adott helyre kössünk. A koherens esetekben nem volt szükségünk véletlen szám generálásra. Itt az identitás kapuk helyett $R_z(\varphi)$ kaput raktunk be a megfelelő helyre, ahol R_z a z tengely körül φ szöggel forgató kapu. Hibás ancilla kiolvasással a kód Qiskites részén nem foglalkoztunk. Utólag generáltuk le a hibás pozíciókat és módosítottuk aszerint az adott hely értékét. A már potenciális hibákkal ellátott szindrómából pedig legyártottuk a megfordulásmátrixot.

5.2. Dekódolás és javítás

A mérések eredményeinek ismeretében meg kell oldanunk a dekódolási problémát, amit a már ismertetett minimális súlyú tökéletes párosítás probléma szerint tettünk meg.

Stabilizátor kódok ilyen módon történő dekódolása bevett gyakorlat, ezért erre is elérhető python szoftver csomag a Pymatching [11].

A Pymatching általános stabilizátor kódra íródott, így először meg kell neki adnunk a stabilizátor generátor elemeit. Esetünkben ez kétszer négy (X és Z stabilizátorokon egymástól függetlenül dolgozunk) kilenc hosszúságú 0-1 sorozat, mely megadja melyik kvantumbiteken nem hat triviálisan az adott stabilizátor. A Pymatching számára emellett biztosítanunk kell a vertikális és horizontális élek súlyát. A súlyok és a stabilizátorok ismeretében a Pymatching.Matching() függvénybe betáplálva a megfordulás mátrixot, az visszaadja a nekünk szükséges javító operátorokat.

5.3. Szimuláció

A szimulációk során a célom az egyes hibacsatornák memória romboló hatásának a vizsgálata volt. Kvantumos állapotok hasonlóságát jellemző mennyiség a kvantumos hűség, aminek átlagos értékével a felületi kód rezisztivitása egy hibacsatornával szemben jól jellemezhető

$$F = |\langle \psi_{kezdeti} | \psi_{vég} \rangle|^2. \quad (5.3.1)$$

Az egyes csatornák és azok különböző paraméterei mellett, az átlagos hűséget Monte-Carlo módszerrel határoztuk meg. Ebből később képeztük a logikai hiba valószínűségét az alábbi definíció szerint:

$$P_{Logikai} = 1 - F \quad (5.3.2)$$

A logikai memória kiindulási állapota tetszőleges lehet, ezért a kiinduló állapotokat egyenletesen kell mintavételeznünk a Bloch-gömb felszínéről. Az általunk vizsgált csatornában az átlagos hűség meghatározásához a Haar-mérték szerinti mintavételezés helyettesíthető 6 pontra vett átlaggal [12]. Ez a hat pont az X, Y és Z sajátállapotok. Előzetes hipotézisünk volt, hogy a hibacsatorna asszimmetriájából adódóan $\bar{X}$ -et okozó hiba nem történhet, illetve terjedés útján sem következhet be. Így a csatornák a $\bar{Z}$ sajátállapotokat nem tudják megváltoztatni. Továbbá, a másik négy pont esetében szimmetria okokból a hatás azonos kell legyen, ezért elegendő egyetlen pontra futtatni. Ezt később az eredmények meg is erősítették.

Minden hibacsatorna és azok különböző paraméterei mellett a hat pontból, mint kiinduló állapotból legyártottuk a Qiskit-es kód segítségével a trajektóriákat, a javítás előtti végállapotot és a mérési eredményeket pedig mentettük. A dekódolást és a javítást nem végeztük el, mert a dekódoláshoz nem tudtuk az optimális vertikális és horizontális súlyokat.

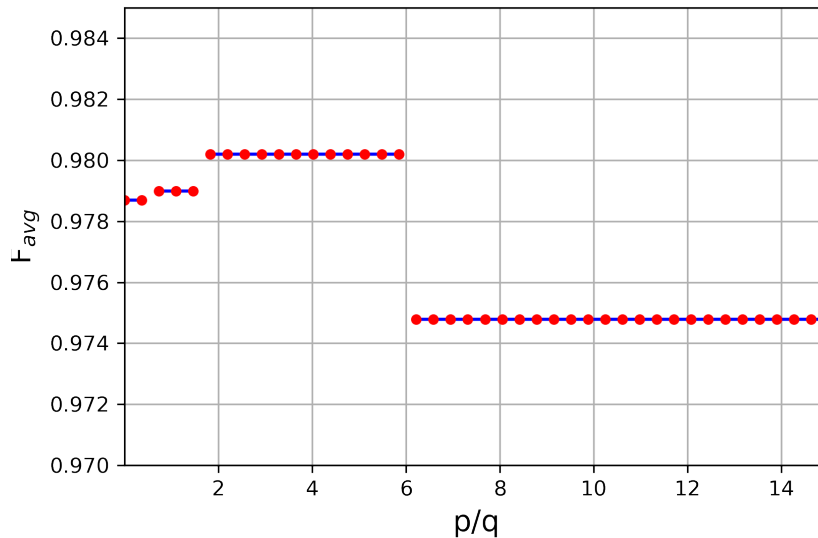
A Pymatchingben a vertikális és horizontális súlyokat két névleges hibavalószínűség p és q alapján képezi.

$$w_{horizontális} = \log \left(\frac{1-p}{p} \right) \quad (5.3.3a)$$

$$w_{vertikális} = \log \left(\frac{1-q}{q} \right) \quad (5.3.3b)$$

A dekóder p és q paraméterek arányának változtatásával megkerestük azt a súlyarányt, amivel a legmagasabb átlagos hűség értéket kaptuk a mintán. Itt nem volt szükségünk minimum kereső algoritmusra, mint például a gradiens módszer, mivel a hűség értékek a $\frac{p}{q}$ arány széles tartományain azonosak voltak. Így csupán a legjobb tartományból egy pontot kellett megtalálnunk. Az azonos eredményt adó tartományok az összes súlyarány

esetében az összes szindrómát ugyanúgy értékeli ki. A szintugrások pedig azokon a helyeken vannak, ahol néhány mérési eredménynél más hipotézis a megoldása a minimális súlyú tökéletes párosításnak.



5.3.1. ábra. Átlagos hűségre (egy hibacsatorna egy adott hibarátájánál) a vertikális és horizontális élek különböző súlyaránya mellett lépcsős függvényt kapunk. Ezt az ábrát a hibátlan kapuk melletti inkohereus Z hiba $p_{ch} = 0.0233$ paraméterére kapott adathalmaz kiértékelése során kaptuk.

A számításainkat a Wigner felhőn végeztük, minden beállítás esetében pontonként 1666 trajektóriát képeztünk, összesen ≈ 10000 darabot. Ez inkohereus csatornák esetében ≈ 16 óra futást vett igénybe. Míg koherens esetben, a forgatások miatt valamelyest nagyobb a számítási kapacitás, így ott a trajektóriák legyártása ≈ 30 órába telt.

5.4. Egy kvantumbit teljesítménye

A különböző csatornákat nem csak egymással hasonlítjuk össze, hanem egyetlen különálló kvantumbittel is. Így válik láthatóvá, hogy jobban teljesít-e a memória, ha azt felületi kódban tároljuk. Az egyetlen kvantumbit három felületi kód ciklus időtartamának megfelelő idejéig van kitéve a hibacsatornának. A harmadik körben viszont a felületi kódon csak a legelső szakaszban lehet hiba, ezért konzisztencia miatt azt mondjuk, hogy az egyetlen kvantumbiten is 625 ns névleges ideig hat a hibacsatorna. Koherens és inkohereus hibára a korábbi egymáshoz igazítás miatt az átlagos hűség azonos. A p_{ch} függvényében könnyen kiszámíthatjuk az hűség értékeket a hat pontra.

$ \psi\rangle$	$P_{logikai}(p_{ch}) = 1 - F(p_{ch})$
$ 0\rangle$	0
$ 1\rangle$	0
$ +\rangle$	$2\frac{1}{12}p_{ch}$
$ -\rangle$	$2\frac{1}{12}p_{ch}$
$\frac{ 0\rangle+i 1\rangle}{\sqrt{2}}$	$2\frac{1}{12}p_{ch}$
$\frac{ 0\rangle-i 1\rangle}{\sqrt{2}}$	$2\frac{1}{12}p_{ch}$

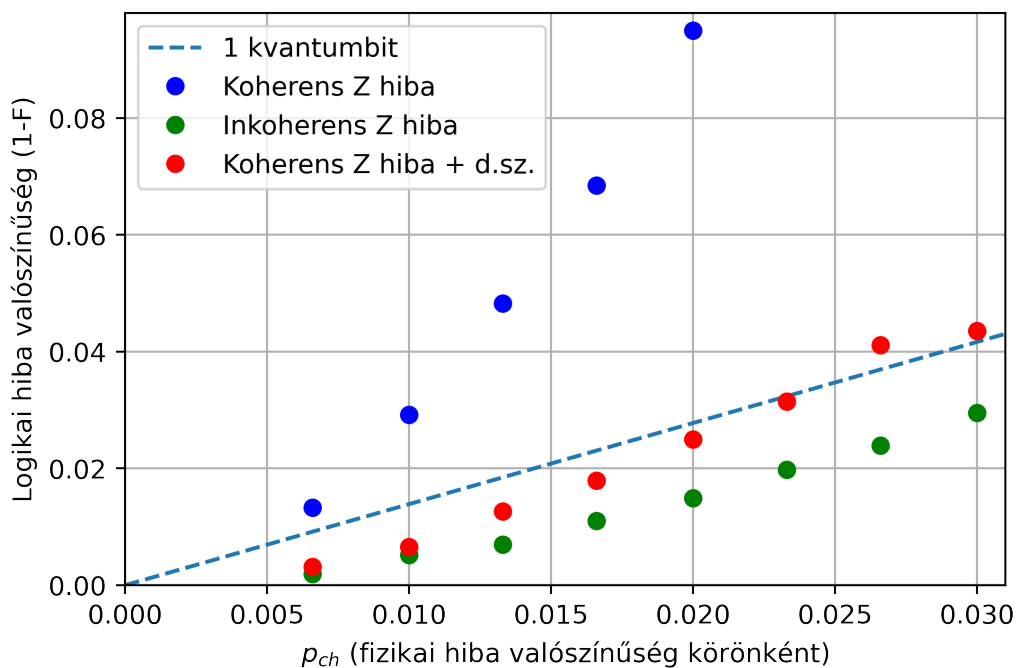
2. táblázat. A logikai hiba valószínűsége egy kvantumbiten a hat kiindulási állapot esetében a koherens- és inkohere ns-Z hibacsatorna hatására

$$P_{logikai\hat{a}tlagos} = 1\frac{7}{18}p_{ch} \quad (5.4.1)$$

6. Szimulációs eredmények

6.1. Z hibák tökéletesen kivitelezett kvantumos kapuk mellett

A tökéletesen kivitelezett kaput feltételező hibacsatornák esetében három esetet vizsgáltunk, az (i) inkohereus Z hibákat, illetve a (ii) koherens Z hibákat dinamikus szétcsatolással és (iii) anélkül. A kiszámolt logikai hiba valószínűségeket a ciklusonkénti hibavalószínűség ($p_{ch} = r\Delta T_{ciklus}$ vagy $p_{ch} = \sin^2(\frac{\varphi\Delta T_{ciklus}}{2})$) függvényében a 6.1.1. ábrán ábrázoltuk. Megjelenítettük az egyetlen kvantumbitre jellemző értéket is, hogy szemléltessük a hibajavítás jelentőségét.



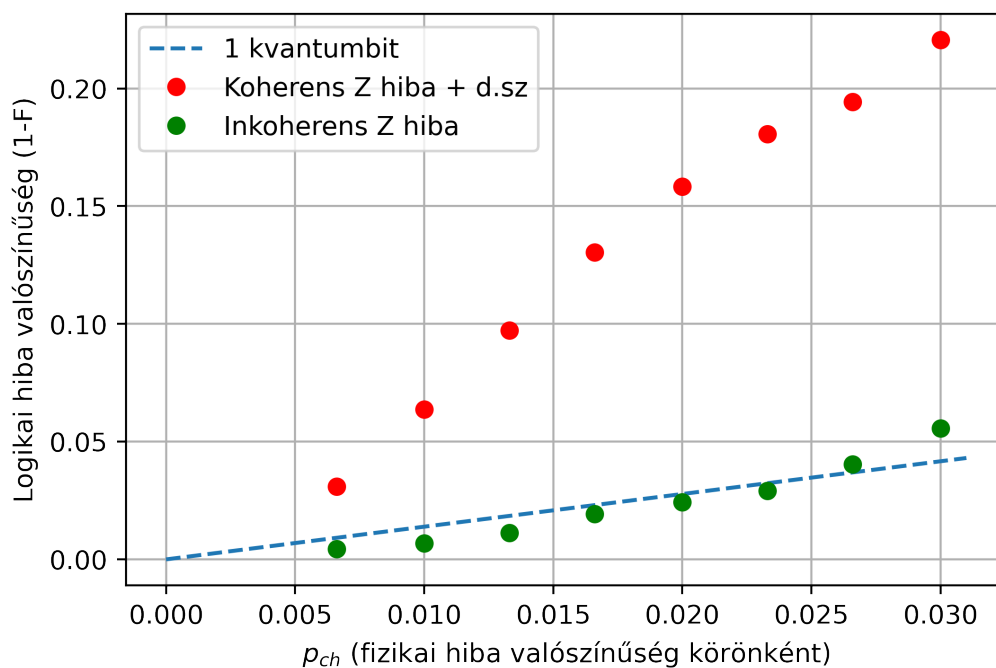
6.1.1. ábra. Az ábrán a logikai hibavalószínűsége látható az egy ciklusidő alatti hibavalószínűség a függvényében, inkohereus Z hiba esetén (zöld), koherens Z hiba esetén (kék) és koherens Z hiba esetén dinamikus szétcsatolással (piros). (Tökéletesen működő kvantumos kapukat tartalmazó konstrukció)

Jól látható, hogy a koherens hibacsatorna ellen sokkal kevésbé rezisztív a felületi kód, mint a Pauli Z hibákkal. Dinamikus szétcsatolással is szigorúan nagyobb hibákat kapunk, mint az inkohereus esetben. A dinamikus szétcsatolás drasztikusan javítja a felületi kód ellenállását koherens típusú hibákkal szemben. Ez alapján arra következtethetünk, hogy a memóriánkra elsősorban az a veszélyes intervallum, amikor kiolvassuk az ancilla kvantumbiteinket. A jelenleg elérhető hardvereken ez az intervallum még hosszabb, ami viszont az inkohereus hibák valószínűségét növeli. Ilyen hibacsatornák esetében a hibajavítással és a dinamikus szétcsatolással ténylegesen jobb hatásfokot tudtunk elérni, mint egyetlen kvantumbittel. Ugyanakkor meg kell jegyezni, hogy a vizsgált modell aszimmetrikus volt és nem tartalmazott hibás CNOT kapukat, amelyek igazán megnövelik a szomszédos kvantumbiteken történő hibák bekövetkezésének valószínűségét. A

szomszédos kvantumbiteken történő hibák esetén gyakrabban tudunk dekódolási hibát végezni, amivel a logikai hibaráta nőhet.

6.2. Z hibák hibával terhelt kvantumos kapuk mellett

A másik konstrukció esetében jóval nagyobb zaj éri a kódot, így előzetesen is már nagyobb logikai hibarátaikat vártunk. A hibátlan kapuknál láttuk, hogy dinamikus szétcsatolás nélkül a koherens esetben a logikai hiba valószínűség nagyon gyorsan növekszik, ezért ebben a konstrukcióban már csak dinamikus szétcsatolással vizsgáltuk a koherens esetet.



6.2.1. ábra. Az ábrán a logikai hibavalószínűsége látható az egy ciklusidő alatti hibavalószínűség a függvényében, inkoherens Z hiba esetén (zöld) és koherens Z hiba esetén dinamikus szétcsatolással (piros). (Tökéletesen működő kvantumos kapukat tartalmazó konstrukció)

Az eredmény hasonló az előző esethez. A koherens hibák ebben az esetben is veszélyesebbek, mint az inkoherensek. Ugyanakkor ebben a konstrukcióban a koherens esetben sokkal gyorsabban romlik a felületi kód rezisztivitása és az általunk vizsgált tartományban már nem is teljesít jobban, mint egyetlen kvantumbit. Alacsonyabb hibarátaik esetén a számítási költség jelentősen megnő, ekkora adatmennyiséggel pontjaink nagy szórásúak lettek volna. Az inkoherens esetre is jóval magasabb értéket tapasztalunk, ennek köszönhetően láthatóvá is vált a pszeudohatár ("pseudothreshold"), ami az a pont, ahol az egykvantumbit már jobban teljesít a hibajavító kódnál.

7. Konklúzió

Munkánk során láthattuk, hogy a Qiskit szoftverfejlesztő csomag alkalmas eszköz a 17 kvantumbites felületi kódon való Pauli- és koherens hibacsatornák állapotvektor szimulációjára. Monte-Carlo módszerrel meghatároztuk koherens és inkoherens Z hibákat tartalmazó hibacsatornák a logikai hibarátaját a $d=3$ kódtávolságú felületi kódon. Azt mutattuk meg, hogy kvantum áramköri szinten a koherens hibák ellen kevésbé ellenálló a felületi kód. Amennyiben a kvantum hardveren a környezeti zaj a koherens zaj modellel jellemezhető, akkor a dinamikus szétcsatolás módszere jelentősen javíthatja a memória élettartamát.

Hivatkozások

- [1] Eleanor G Rieffel and Wolfgang H Polak. *Quantum computing: A gentle introduction*. MIT Press, 2011.
- [2] A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.
- [3] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, et al. Strong quantum computational advantage using a superconducting quantum processor. *Physical review letters*, 127(18):180501, 2021.
- [4] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.
- [5] Rajeev Acharya, Igor Aleiner, Richard Allen, Trond I Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Juan Atalaya, Ryan Babbush, et al. Suppressing quantum errors by scaling a surface code logical qubit. *arXiv preprint arXiv:2207.06431*, 2022.
- [6] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. In *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, volume 68, pages 13–58, 2010.
- [7] Dan Browne. Topological codes and computation. <https://sites.google.com/site/danbrowneucl/teaching/lectures-on-topological-codes-and-quantum-computation>, 2014.
- [8] Yu Tomita and Krysta M Svore. Low-distance surface codes under realistic quantum noise. *Physical Review A*, 90(6):062320, 2014.
- [9] Andre Van Rynbach, Ahsan Muhammad, Abhijit C Mehta, Jeffrey Hussmann, and Jungsang Kim. A quantum performance simulator based on fidelity and fault-path counting. *arXiv preprint arXiv:1212.0845*, 2012.
- [10] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
- [11] Oscar Higgott. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing*, 3(3):1–16, 2022.
- [12] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Physical Review A*, 80(1):012304, 2009.