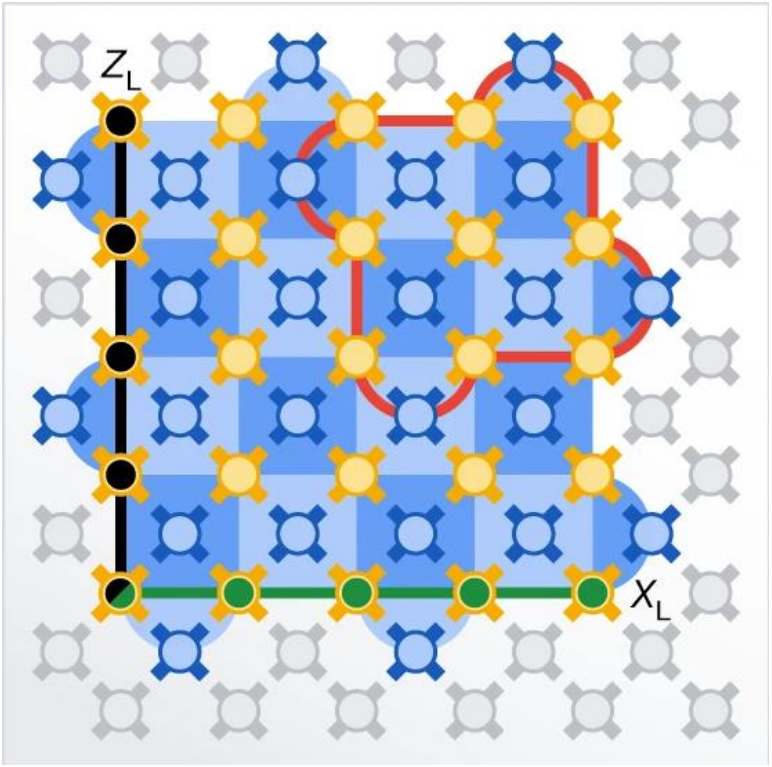
Interaktív animáció a kvantumos hibajavításról: a felületi kód

- ***Zanathy Marcell***
- **Konzulensek:** ***Márton Áron, Dr. Asbóth János***
- **Fizikatanár:** ***Dr. Jarosievitz Beáta***



Tartalomjegyzék:

Kvantumbitek működése:

- **Kvantumbitek állapota:**

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle, \text{ ahol } \alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 = 1$$

- **A klasszikus bitekkel ellentétben a kvantumbitek a bázisállapotok bármely *szuperpozícióját* felvehetik.**
- **Kvantumbitek *mérés*énél a rendszer beugrik egyik *bázisállapot*ába.**

→ **A mérés eredménye: *+1* vagy *-1*.**

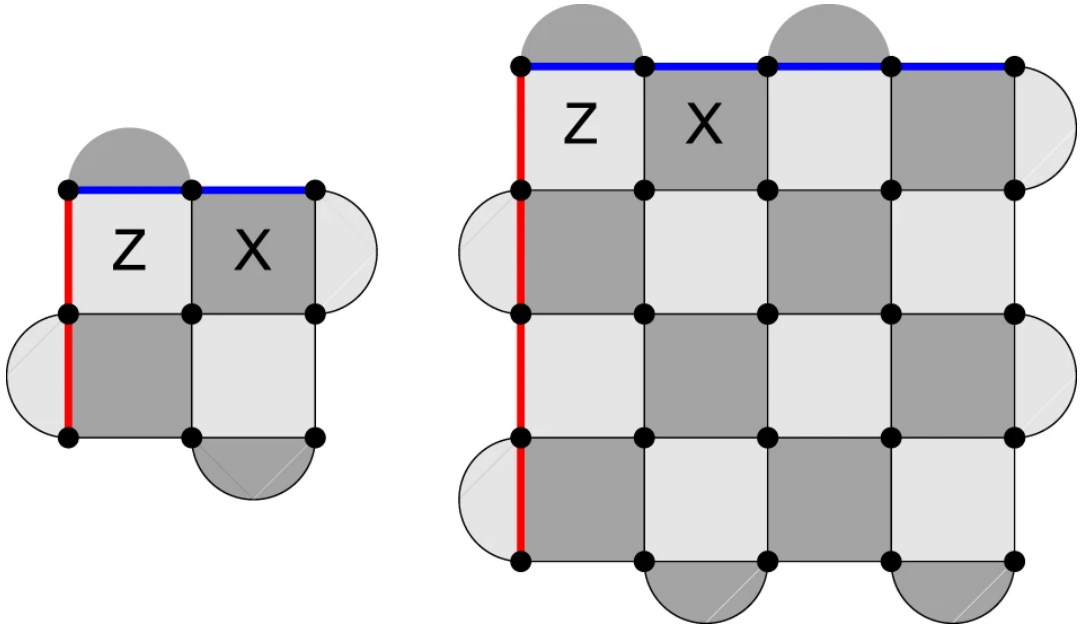
$$P_{+1} = |\alpha|^2, \quad P_{-1} = |\beta|^2$$

Hibatípusok:

- $X : |0\rangle \rightarrow |1\rangle,$
  $|1\rangle \rightarrow |0\rangle$
- $Z : |0\rangle \rightarrow |0\rangle,$
  $|1\rangle \rightarrow -|1\rangle$

Hogyan lehetséges a kvantumbitek hibatűrő tárolása?

- ***Logiaki állapot*ok: 9, illetve 25 kvantumbit szuperponált, összefonódott állapota, melyben a kvantuminformációt tároljuk**
- **A *stabilizátor*ok úgy vannak definiálva, hogy minden kvantumbiten tudjuk korrigálni mindkét hibatípust a *felületi kód*on.**

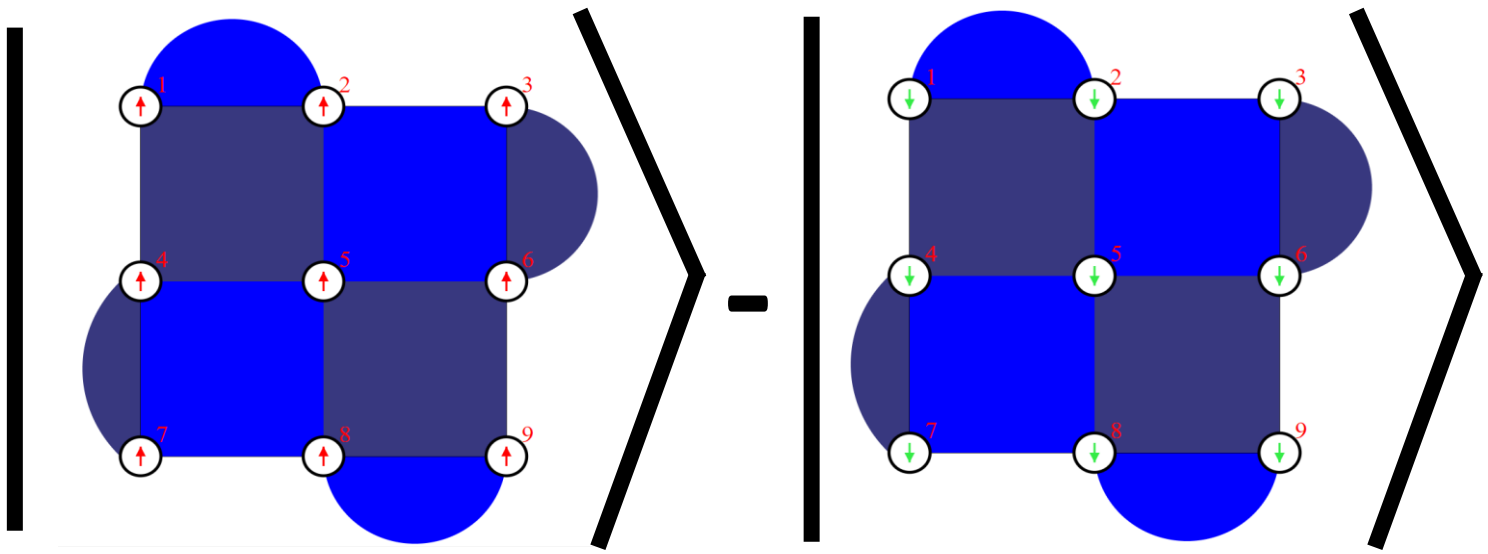


Az animáció megjelenése:

- **Időben állandó szuperponált állapot megjelenítése**

$$|\psi\rangle = \frac{1}{\sqrt{2}}|000000000\rangle - \frac{1}{\sqrt{2}}|111111111\rangle$$

=



Lefutási idő: *2 sec*

= ***1 sec*** + ***1 sec***

ChatGPT, mint asszisztens: minőség-ellenőrzés

- Ismeretlen JavaScript nyelv

  $\Rightarrow$ ChatGPT felhasználása

- D3.js könyvtár

- Kód hossza: 655 sor

- Prompt engineering

Módszertani megjegyzések:

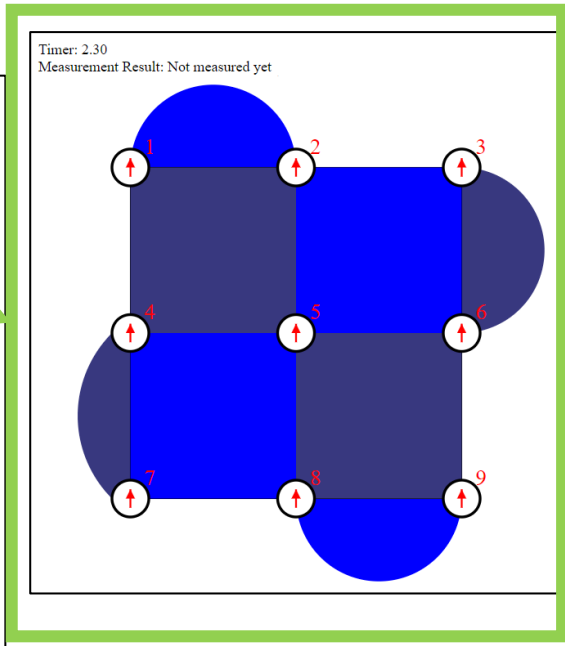
- Némi programozási és az elkészítendő program tárgyára vonatkozó *előismeret*
- *Angol nyelvű* kommunikáció
- *ChatGPT-4* használata (fizetős - jobb teljesítmény)

Az animáció elkészítésének menete:

Grid Size (row, col): 3,3
Quantum State: 1|000000000>
Total Duration: 4

Start Animation
Stop Animation

1. A felületi kódot megjelenítő *rács*



Enter qubit numbers, e.g., 1,2 Apply Pauli X

Enter qubit numbers, e.g., 1,2 Apply Pauli Z

Stabilizers:

- SX_1: $X_1X_2X_4X_5$
- SZ_1: Z_1Z_2
- SX_2: X_3X_6
- SZ_2: $Z_2Z_3Z_5Z_6$
- SX_3: X_4X_7
- SZ_3: $Z_4Z_5Z_7Z_8$
- SX_4: $X_5X_6X_8X_9$
- SZ_4: Z_8Z_9

2. *Operátorok és stabilizátorok* hattatás

Apply Logical X | Apply Logical Z

Apply Sx1 | Apply Sz1 | Apply Sx2 | Apply Sz2 | Apply Sx3 | Apply Sz3 | Apply Sx4 | Apply Sz4

Measure Sx1 | Measure Sz1 | Measure Sx2 | Measure Sz2 | Measure Sx3 | Measure Sz3 | Measure Sx4 | Measure Sz4

Measure Logical Z Operator | Measure Logical Z Operator

Timer: 1.00
Measurement Result: -1

3. *Mérés* ⇐ a folyamat legbonyolultabb része

MA

Now the time each basis vector should be displaied on the screen should be calculated with the following formula (as I gave it before): t = |c|^2*n/ |s|^2 where „t" is the time each given basis vector is displaied, „n" is the duration (in seconds) of the total animation of every quantum state that is displaied - thats the value that the user gives before the animation, „s" is the sum of every coefficent and „c" is the coefficent corresponding to the basis vector that is currently displaied on the screen.

- **Konkrét *képletek*, *összefüggések* megadása, ahol ismert.**

MA

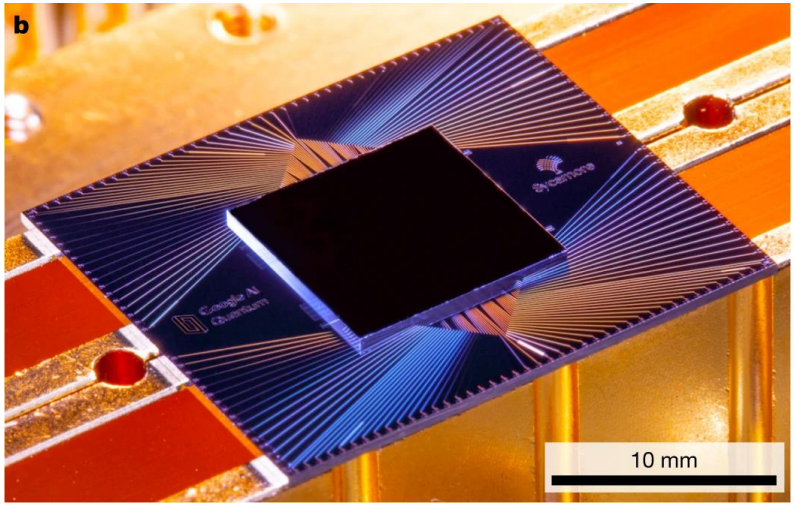
Calcualte the following state: c=a+b where a is the displayState and b is the displayState (a) after applying the X1 operator to it. Then count the number of different basis vectors in c. If some basis vector cancels each other out in this new state (a+b) then do not count it, and you have identical basis vector in the new state (a+b) count it as many times as they appear in the new state (a+b). After this you get the first number (c) you have to work with.

- **Hatékonyabb eredmény elérése érdekében fogalmazzuk meg *egyszerűbb szinten* a komplex utasításokat, *analógiák*ra redukálva azokat.**

$$P_{\pm 1}^{O} = \frac{1 \pm O}{2}$$

Összefoglalás:

- **Dolgozatomban egy *interaktív animáció*t készítettem a felületi kódról**
- **A kód teljes hossza: *655 sor***
- **JavaScript nyelven íródott a *ChatGPT* bevonásával**



Fontosabb források:

- A. Kitaev, „Fault-tolerant quantum computation by anyons,” Annals of Physics, vol. 303, pp. 2–30, jan 2003.
- Pesah, „An interactive introduction to the surface code,” 2023. Accessed: 2023-10-30 https://arthurpesah.me/blog/2023-05-13-surface-code/
- E. G. Rieffel and W. H. Polak, Quantum computing: A gentle introduction. MIT Press, 2011.

Kérdések:

1. **Véleménye szerint összességében gyorsabb volt-e a ChatGPT-vel megíratni az interaktív animációt, mint amennyit idő alatt Ön saját maga meg tudta volna írni mindezt?**

- Egy JavaScript képzés körülbelül 120 órát vesz igénybe (mindennel együtt). Ez az én esetemben legyen inkább *150 óra*. A kód megírásának ideje a ChatGPT-vel hasonló körülbelül: *144 óra*. (Egy héten körülbelül 10-12 órát dolgozatam a kód megírásán három hónapon keresztül.) Ezt a csekély különbséget viszont *nem tartom kifejezetten relevánsnak* a munka természetét figyelembe véve. Természetesen, ha valaki ismer egy adott programozási nyelvet, akkor a munkafolyamat nagymértékben átalakul (pl.: rutinfeladatok).

2) Milyen módszereket használt a ChatGPT által megírt program ellenőrzéséhez? (Elsősorban a kvantummechanikai számítások ellenőrzéséhez, amelyek értelmezése kapcsán a ChatGPT nem megbízható.)

- Először a számításokat *papíron* végezetem el, majd az eredményeket összevetettem a képernyőre kiíratott kvantumállapottal, melybe a szimulált felületi kód került a művelet elvégzése után (esztétikai okokból a végső animációba nem került bele ez a kiírás). Továbbá, számos olyan *példán* keresztül teszteltem a kódot, amelyek alapján nagy magabiztossággal kizárhatóak az esetleges hibák. Illetve a *megíratott kód* átlátható volt olyan mértékben, hogy a számítási logika részletei világossá váljanak számomra.