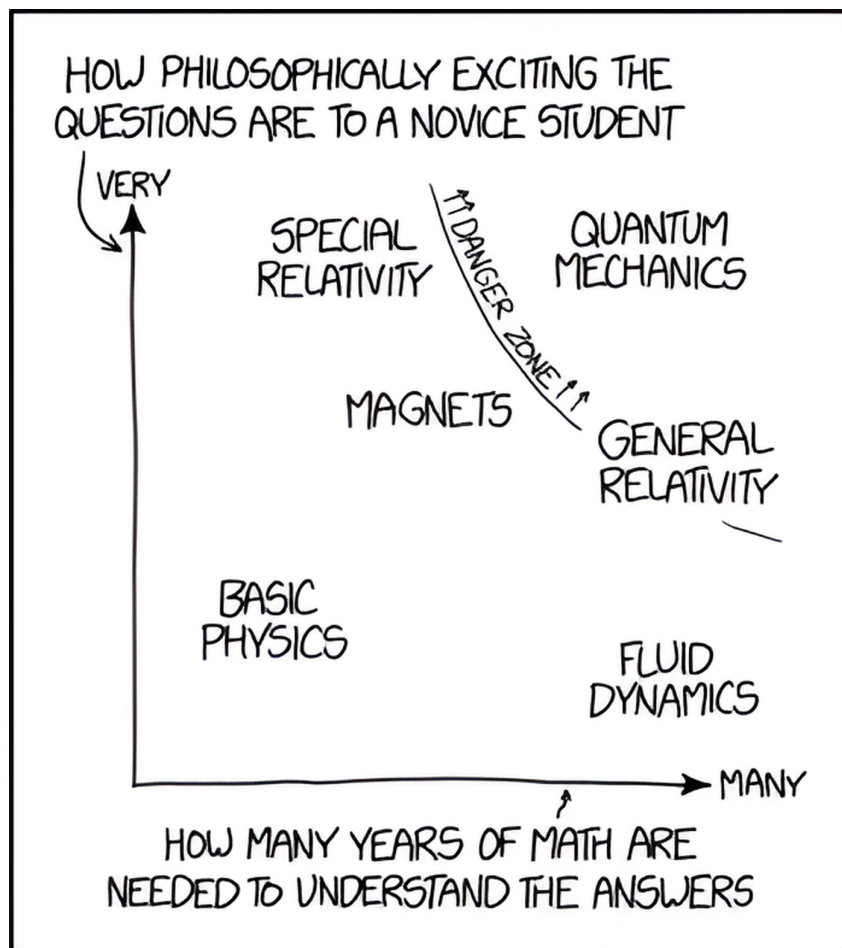# Quantum Cellular Automata

NIUS Physics (Batch 20)
Mid–Term Report
Camp: 20.2

PIYUSH KUMAR SINGH*

IISER KOLKATA

SAMBUDDHA SANYAL†

IISER TIRUPATI

January 26, 2024

*pks22ms027@iiserkol.ac.in
†sambuddha.sanyal@iisertirupati.ac.in

Acknowledgement

Contents

1 Introduction

Cellular automata are discrete mathematical models used to simulate complex systems. John von Neumann first introduced the concept [7] in 1966. In 1970, an article written by Martin Gardner [6] introduces us to a compelling use case of this abstract concept, "The Game of Life," invented by J. H. Conway.

In a conference in 1982, R. P. Feynman expressed his view (or dream) of using 'Quantum Physics' in computers [3] to simulate complex physical systems using the idea of density matrix (proposed by Neumann in 1955 [9]); building on his idea Feynman introduced the world to a weird collaboration named "Quantum Cellular Automata" (QCA) in an article published in 1986 [4].

And there are multiple reasons why we should care about QCAs. First, this is part of a broad field of 'Quantum Information Processing,' any development in QCA may help us understand how we can harness quantum properties for computation. Second, the classical systems exhibit some new emergent behaviors, so we want to know if we can also have these emergent behaviors in QCAs.

Also, in the later part of the discussion, we will see the application of these 'discrete' quantum systems (in the form of "Random Quantum Circuits"[5]), observing some of the various emergent properties like:

(a) Fidelity,
(b) Information spread (encoded as *Operator Spread*).

§1.1 Cellular Automata

A cellular automaton (CA) is a group of "coloured" cells on a predetermined-shaped grid that follows a set of rules depending on nearby cell states to evolve over many discrete time steps. After that, the rules are applied repeatedly for as many time steps as needed.

It turned out that reaction-diffusion systems, biological pattern development, fluid flows, and traffic models are only a few real-world uses for CAs. Lattice gas cellular automata are a specific type of CA used to simulate fluid dynamics. The Navier-Stokes equation can be obtained by selecting the appropriate model. Lattice Boltzmann models have taken their place, using continuous functions at the lattice locations rather than discrete variables. More aspirationally, CAs have been proposed as discrete physics models with many desirable characteristics, such as dynamical homogeneity and localization. But, while this is a fascinating idea, physics is ultimately quantum, and CAs cannot adequately represent, e.g., Bell inequality violation, which occurs from quantum entanglement.

Before giving a formal definition, let's look at an example:

Example 1 (Rule Number 110)

Consider a one-dimensional array of bits (*i.e.* cells having two states, namely '0' or '1'). The update rule of a bit is dependent on the initial state of that bit as well as the state of its neighbouring bits. The rule can be summarized as follows:

Initial state of a bit and its neighbours	111	110	101	100	011	010	001	000
New state of middle bit	0	1	1	0	1	1	1	0

Table 1: Update rule for a *one*-dimensional CA

We named this update rule as `Rule 110` because if you treat the last row of the table as a binary number, it converts to 110 in the decimal system. This CA can simulate a Turing machine efficiently (with only polynomial overhead).
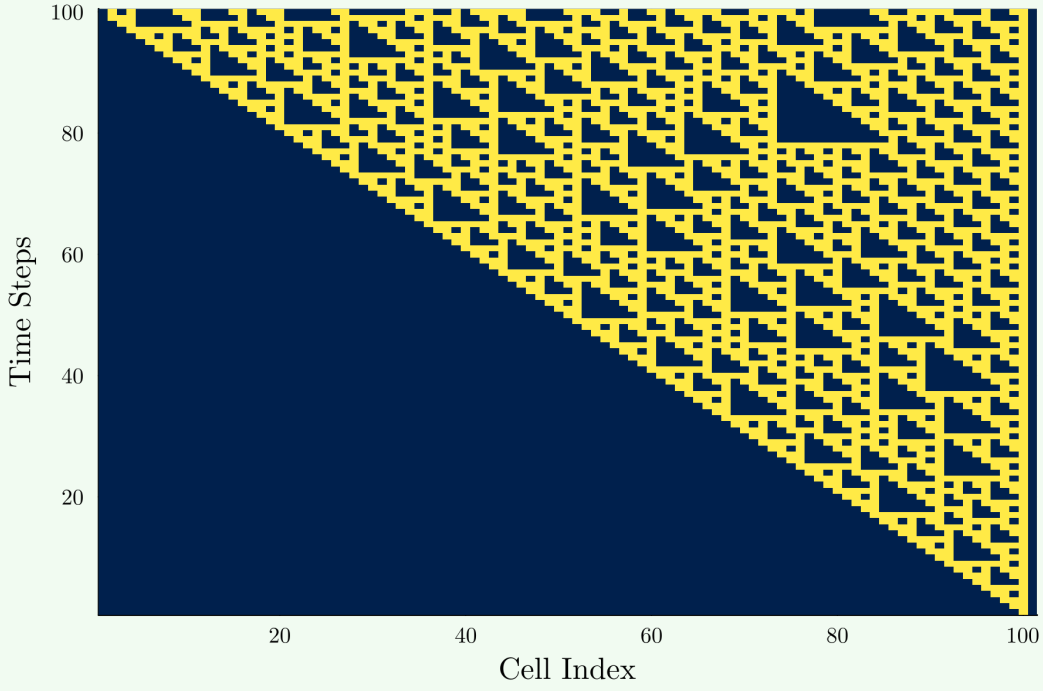

Figure 1: An example of a CA evolution for the rule 110 CA, with time going up. Bits with value 1 are represented by the yellow squares, while blue squares represent 0.

Definition 2 (Cellular Automata):
A Cellular Automaton is a 4-tuple $(L, \Sigma, \mathcal{N}, f)$ consisting of a d-dimensional lattice of cells indexed by integers, $L = \mathbb{Z}^d$, a finite set Σ of cell states, a finite neighbourhood scheme $\mathcal{N} \subseteq \mathbb{Z}^d$, and a local transition function $f : \Sigma^{\mathcal{N}} \to \Sigma$.

To better understand this local transition function f, consider a cell $x \in L$. This function takes the state of the neighbors of x as the argument, which is indexed by the set $\mathcal{N}$ at the current time $t \in \mathbb{Z}$ to spit out the state of cell x at time $t + 1$.

This observation points us towards two important properties of cellular automata that will help us understand why CAs are very helpful in simulating certain systems:

- cellular automata are **space-homogeneous** *i.e.* the local transition function performs the same function at each cell.

- Also, cellular automata are **time-homogeneous** *i.e.* the local transition function does not depend on the current time t.[1]

We can define the current state of the lattice (or CA) as a *configuration* $C \in \Sigma^L$, which has the information about the state of each cell at time t. Now, using this idea, we can define a 'global' transition function $F : \Sigma^L \to \Sigma^L$, which acts on the entire latter rather than on individual cells and spits out another configuration C' for the lattice at time $t + 1$.

Remark 3 (Revisiting Rule 110). Observe the example 1; we can fit this scenario in the definition. In this case $L = \mathbb{Z}$, $\Sigma = \{0,1\}$, for i^{th} cell $\mathcal{N} := \{i-1,\ i,\ i+1\}$ and f is defined according to the update rule specified in table 1.

---

[1] We will review these properties in detail in sections 1.1.2 and 1.1.3

§1.1.1 Characteristics of Cellular Automata

In the definition and example, we have used a few terms; now we will define them (and these will be the main characteristics of cellular automata):

Discrete Space-Time: In our discussion, we have defined the lattice (L) to be d-dimensional integer space, which makes our area of interest a discrete space. Our evolution to the next state is also discrete since we evolve our states from $t \to t+1$ (or predefined time steps).

Finite State Set: The set Σ is finite *i.e.* every cell has only finitely many states of being in. Generally, in classical systems, we have state set $\Sigma = \{0, 1\}$, also called 'binary cellular automata.'

Cell/Cell-state: Every space-position vector $i \in L = \mathbb{Z}^d$ identifies a cell which contains a cell-state $c_i \in \Sigma$.

Configuration: A configuration C is the concatination of all cell-states over the entire lattice L at a given time t which means that every configuration is an element of set Σ^L.

Local Transition Function: We know time is also discrete in our system (with a predefined time step). We use a local transition function f that defines the state of every cell after each time evolution. We refer to this function as a local function, as it only takes the states of the 'neighboring' cells as an argument.

Neighbourhood Scheme: As we have defined, the outcome of our local function at any cell i depends on the set of cell-states of neighboring cells. This set is defined by the neighborhood scheme $\mathcal{N}$ on cellular automata.

Global Function: For a given lattice L (with a defined neighborhood scheme), the local transition function f imposes a global transition function F on the set of all possible configurations. This global function determines the *space/time* behavior of the cellular automaton on an initial configuration.

§1.1.2 Locality

In physics, locality implies we are talking about the PRINCIPLE OF LOCALITY, which states that an object is influenced directly only by its immediate surroundings. If a theory that includes this principle is said to be a "local theory." This idea of locality evolved from the classical field theory; building upon this idea, if an event at a point is cause for an effect at another point, a wave or a particle must travel through space-time between the two points, carrying the influence.

Einstein postulated that causal influence couldn't be transmitted between two points faster than the speed of light c, in the Special Theory of Relativity, which means that if two points are spacelike separated, they can't influence each other.

In the context of cellular automata, we know that the state of a cell is only influenced by the states of finite neighbors (decided by neighborhood scheme). Since the state of any cell is influenced by its surroundings, we can say that the cellular automata theory is local.

> Remark 4 (Relation between locality and relativistic causality). Causality means an effect can only occur from a cause in the past light cone of that event. Similarly, a cause can only affect an event in the future light cone.
>
> The requirement that causal influences cannot propagate faster than the speed of light ensures that physical interactions remain local, implying distant events cannot have instantaneous effects on each other.

§1.1.3 Universality

Universality refers to the idea that certain phenomena or behaviors are independent of the specific details of a system, and these phenomena are governed by more basic physical principles or underlying symmetries of the system.

There are multiple scenarios in which we get this universality from the nature:

Quantum Computing: A specific set of quantum gates can efficiently simulate any other quantum circuit, and this set is called the universal gates. So this feature makes quantum computers universal as their behavior is the same, independent of the specific details.

Statistical Mechanics: The distribution of particle velocities in a gas, as described by the Maxwell-Boltzmann distribution, is universal for classical gases, independent of the specific nature of the particles.

Cellular automata are universal by definition as they evolve under a predefined rule (or local transition function) irrespective of their initial state and the time step at which we apply the evolution.

§1.2 Quantum Cellular Automata

R. P. Feynman introduced the idea of Quantum Cellular Automata with a vision that it can be used to simulate 'Quantum Systems' as we don't have a classical analog of many quantum phenomena. In [10], he has proven that QCAs are an alternative paradigm of quantum computers. They have shown QCAs to be universal, implying they could efficiently simulate a quantum Turing machine.

This demonstration showed that QCAs are very important for quantum computers, so many have tried to formulate a theory. But there was an enormous challenge in front of the scientists: after some local function has updated one cell, we will lose the information about the state of the cell at time t as we require it for updating other cells. So somehow, we have to clone the state of this cell to some other cell, but by the **no-cloning theorem**, it is simply impossible.

Werner and Schumacher (in [8]) gave an axiomatic definition for QCAs to avoid these theoretical challenges. Farrelly in [2] restated the definition as follows:

> **Definition 5 (Quantum Cellular Automata):**
> A Quantum Cellular Automaton is a spatial lattice $L = \mathbb{Z}^d$, with a quantum system on each lattice point $x \in L$ and evolves over discrete timesteps via some unitary operators that are locality preserving.

If we look at the definition, it becomes evident that locality here refers to **relativistic causality**. As we have defined our system on a discrete spacetime, we have assumed the maximum propagation speed of information. This definition guarantees locality; combined with unitarity, we get a QCA.

> We claim to simulate physics with these discrete spacetime systems in this discussion. However, all theoretical physics works with continuous spacetime.
> Can we discretize whole physics is a nontrivial and exciting question on its own.

§1.2.1 An example of QCA [2]

For a typical quantum system (not a QCA), the dynamics is determined by the Schrödinger equation. However, our systems are defined on discrete spacetime, so there is no corresponding Schrödinger equation. So, we work with unitary operators that evolve the state every time step.

Take a discrete line of qubits with position labelled by $x \in \{0,\ 1,\ 2, \ldots, N-1\}$, where N is even. Consider the 'depth-two' circuit of unitary operators; say, the unitary operator V acts on two qubits at a time named $2n$ and $2n-1$. The unitary operator W acts on two qubits named $2n+1$ and $2n$ (see fig. 2).

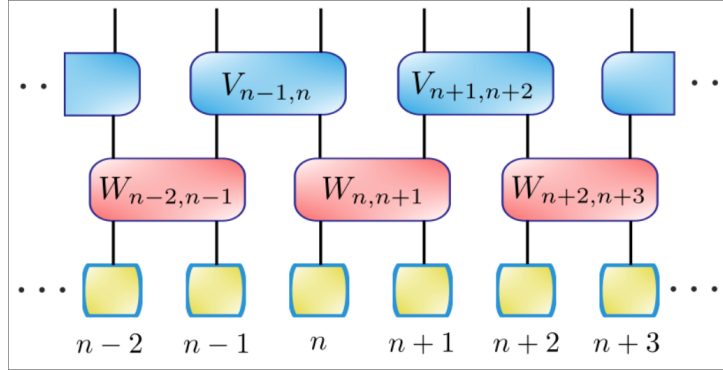**Figure 2:** A simple example of a QCA unitary.

The yellow boxes represent qubits, while the red and blue rectangles represent the two-qubit local unitaries that implement the dynamics over one timestep.
So, the unitary operator for this 'depth-two' circuit will be given by

$$U = \prod_n V_{2n,\ 2n-1} \prod_m W_{2m-1,\ 2m}$$

The order of application of these unitary operators doesn't matter as they commute.

2 Random Quantum Circuits

Discrete quantum circuits are good examples of QCAs (see section 1.2.1). In [5], they have extended this idea of quantum circuits as QCAs to incorporate an element of randomness into the local unitary gates (or operations). They also included measurements of quantum circuits (which are irreversible and non-local). This kind of quantum circuit is known as "Random Quantum Circuit (RQC)."

This discussion will use a simple discrete-time model for many-body quantum systems. This model's lattice of spins (qubits) evolves by applying local unitary gates and projective measurements. This discrete-time structure can be seen as an example of '`Totterization`' of a continuous-time Hamiltonian evolution $\left[\text{i.e. } e^{i\hat{H}t/\hbar}\right]$, assuming the time-step to be finite (*i.e.* $t \in \mathbb{Z}$), which also implies that energy is not conserved in this approach.

§2.1 Methodology

In this section, we will construct the system used in this analysis. And also go through some of the mathematical tools required to understand "QUANTUM ENTANGLEMENT."

§2.1.1 Defining System

Let a d-dimensional lattice $L = \mathbb{Z}^d$ composed of qubits (or in general q-level systems "`qubits`") which represents a d-dimensional quantum many-body system arranged in a spatially local manner. The discrete-time evolution of such a quantum many-body system by applying local unitary gates defines a quantum circuit. We will restrict our discussion to

    (i) local quantum gates – unitary transformation acting on a few neighbouring qubits,
    (ii) local projective measurements – observations which leave the measured qubit in a state with definite eigenvalue of the measured operator.

§2.1.2 Quantum Entanglement

In 1935 [1], Albert Einstein, Boris Podolsky, and Nathan Rosen published a paper on counter-intuitive predictions that quantum mechanics makes for pairs of objects prepared together in a particular way. In this study, the three formulated a thought experiment (today known as EPR-paradox) to show that "the quantum-mechanical description of physical reality given by wavefunctions is not complete."

> **Definition 6 (Quantum Entanglement):**
> An entangled system is one whose quantum state cannot be separated as a tensor product of the quantum states of its local constituents.

Consider two arbitrary quantum many-body systems A and B, with respective Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$. The composite Hilbert space is a tensor product given by $\mathcal{H} := \mathcal{H}_A \otimes \mathcal{H}_B$. Say system A is in a state $|\Psi\rangle_A$ and B is in $|\Psi\rangle_B$, the state of the composite system is given by $|\Psi\rangle_A \otimes |\Psi\rangle_B$.

> **Remark 7 (Pure and Entangled state).** If the state of the composite system can be expressed in the tensor product of the states of individual systems, then we call this composite state a separable state or `pure state`.
> Not all states are pure states. Let $\{|\mu\rangle_A\}$ be a set of basis for the space $\mathcal{H}_A$ and set $\{|\nu\rangle_B\}$ be a

basis for the space $\mathcal{H}_B$. The most general state in $\mathcal{H}$ is of the form

$$|\Psi\rangle = \sum_{\mu\nu} c_{\mu\nu} |\mu\rangle_A \otimes |\nu\rangle_B \tag{2.1}$$

This state is separable if and only if there exist vectors $\begin{bmatrix} c_\mu^A \end{bmatrix}$ and $\begin{bmatrix} c_\nu^B \end{bmatrix}$ such that $c_{\mu\nu} = c_\mu^A c_\nu^B$, yielding $|\Psi\rangle_A := \sum_\mu c_\mu^A |\mu\rangle$ and $|\phi\rangle_B := \sum_\nu c_\nu^B |\nu\rangle$. And it is inseparable if such vectors do not exist. If a state is inseparable, it is called an '**entangled state**.'

§2.1.2.1 Density Matrix

The density matrix language provides a convenient way to describe quantum systems whose state is unknown.

Definition 8 (Density Matrix):
Suppose a quantum system is in one of several states $|\Psi_i\rangle$, with respective probabilities p_i. We define $\{p_i, |\Psi_i\rangle\}$ an *ensemble of pure states*. The density matrix can be defined as follows for this quantum system
$$\rho \equiv \sum_i p_i |\Psi_i\rangle \langle \Psi_i|$$

Properties of density matrix:

- $\rho^\dagger = \rho$
- $\text{Tr}\,\rho = 1 \quad \Rightarrow \quad \sum_i p_i = 1$
- $\text{Tr}\,\rho^2 \leq 1 \quad$ (equality holds for a pure ensemble)

Reduced Density Function:
Consider a quantum many-body system composed of N qubits and defined by a wavefunction $|\Psi\rangle$, which is bipartitioned into a subset of spins A, with Hilbert space dimension D_A, and its complement $\bar{A}$, with Hilbert space dimension $D_{\bar{A}}$.

Generally, there is no way to associate a pure state to a sub-system A. However, associating a density matrix with this sub-system is still possible. Let $\rho_T = |\Psi\rangle\langle\Psi|$ which is the projection operator onto this composite state. The density matrix for sub-system A is defined as a **partial trace** of ρ_T over the basis $\{|\nu\rangle\}$ of sub-system $\bar{A}$:

$$\rho_A := \sum_\nu \left(\mathbb{I}_A \otimes \langle\nu|_{\bar{A}} \right) \left(|\Psi\rangle\langle\Psi| \right) \left(\mathbb{I}_A \otimes |\nu\rangle_{\bar{A}} \right) = \text{Tr}_{\bar{A}} |\Psi\rangle\langle\Psi| \tag{2.2}$$

where, $\mathbb{I}_A$ is the identity operator in Hilbert space $\mathcal{H}_A$. ρ_A is also referred to as *reduced density matrix* of ρ_T on sub-system A, encodes all operator expectation values solely within region A.

§2.1.2.2 Entanglement Entropy

Definition 9 (Entanglement Entropy):
The entanglement entropy measures the degree of quantum entanglement between two sub-systems constituting a two-part composite quantum system.

Continuing the discussion of N qubits system, which has been bipartite into two sub-systems A and $\bar{A}$. There are multiple ways to define entanglement entropy for these two sub-systems; a few of them are listed below:

<u>von Neumann Entropy:</u> For a bipartite system, von Neumann entropy S is defined as follows:

$$S_A = -\text{Tr}(\rho_A \ln \rho_A) = -\text{Tr}(\rho_{\bar{A}} \ln \rho_{\bar{A}}) = S_{\bar{A}} \tag{2.3}$$

<u>n^{th} Rényi Entropy:</u> $S_A^{(n)} = \ln(\text{Tr}\,\rho_A^n)/(1-n)$, and as $n \to \infty$, $S_A^n \to S_A$ *i.e.* the von Neumann entropy.

§2.2 A four-qubit example of RQC

In this section, we will see what a local unitary operator is in light of quantum circuits and what a projective measurement is. Then, we will work out a discrete-time evolution of a four-qubit system under this local unitary operator and projective measurement numerically.

§2.2.1 Basics of Quantum Computer – for a single qubit

For a single qubit, we have a finite state space $\Sigma = \{0, 1\}$, so we can infer that Hilbert space $\mathcal{H}$ for this system is 2-dimensional. The state set can be seen as eigenvalues of a Hermitian operator (to be precise, a Pauli matrix) Z. With these assumptions, we can define the eigenbasis for this Hilbert space $\mathcal{H}$ as

$$\mathbb{B}_Z = \left\{ |0\rangle = |\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \ |1\rangle = |\downarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\} \tag{2.4}$$

This basis set is sometimes referred as "*computational basis.*" Similarly, we can define the other two sets of eigenbasis corresponding to the two remaining Pauli matrices.

$$\mathbb{B}_X = \left\{ |+\rangle = |\rightarrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \ |-\rangle = |\leftarrow\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \right\} \tag{2.5}$$

In general, the time evolution of a quantum system (which is described by a wavefunction $|\Psi\rangle$ at time $t = 0$) under a given Hamiltonian $\mathscr{H}$ which is represented by a matrix $\mathcal{O}$ is provided by

$$|\Psi(t)\rangle = \mathcal{U}(t) |\Psi\rangle, \qquad \mathcal{U}(t) := \exp\left(-i\mathcal{O}\frac{t}{\hbar}\right)$$

For a single qubit, all possible Hamiltonians are 2×2 Hermitian operator, which can be written as a linear combination (over $\mathbb{R}$) of the three "`Pauli Matrices`" and the identity matrix. Define $S = \{\mathbb{I}, X, Y, Z\}$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad \mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.6}$$

Using this construction, we can decompose any Hamiltonian operator $\mathcal{O}$ as

$$\mathcal{O} = \sum_S c_S S = c_{\mathbb{I}}\mathbb{I} + c_X X + c_Y Y + c_Z Z$$

the coefficients in this expansion are real and given by $c_S = \frac{1}{2}\operatorname{Tr}(S\mathcal{O})$. The corresponding unitary can be written as

$$u = \exp\left(-i\sum_S h_S S\right)$$

where, the coefficients h_S are real.

Hadamard Gates: This gate is often referred as "superposition gate", and represented as H in quantum circuits. It maps the computational basis as follows:

$$|0\rangle \xmapsto{H} \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \quad ; \quad |1\rangle \xmapsto{H} \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle$$

The name superposition is evident after looking at the map, as it creates an equal superposition state. From the map itself, we can compute the matrix representation as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.7}$$

Quantum Random Walk:
Consider a quantum system composed of one qubit whose dynamics are governed by unitary gates and projective measurements. If the qubit is in a pure state, then the state is a point on the Bloch sphere, defined by the polarization vector $(\langle X \rangle, \langle Y \rangle, \langle Z \rangle)$. Unitary transformations rotate the state vector on the "Bloch Sphere." A projective measurement, say along Z, causes a stochastic jump to the north or the south pole, depending upon the measurement outcome.

   Due to this randomness, an arbitrary sequence of unitaries and measurements gives a Random Walk on the Bloch Sphere, also called "Quantum Random Walk."

§2.2.1.1 Two qubit system and controlled gates

We define the system's state by tensor product for a pair of qubits. Let the two qubits be indexed as 1, 2, so the Hilbert space of the composite system is $\mathcal{H}_{12} := \mathcal{H}_1 \otimes \mathcal{H}_2 = \mathcal{H}^{\otimes 2}$, keep in mind $\mathcal{H}_1 = \mathcal{H}_2 = \mathcal{H} = \mathrm{span}(|0\rangle, |1\rangle)$. Recall eq. (2.1), any general state of this system $|\Psi\rangle \in \mathcal{H}_{12}$ can be written as

$$|\Psi\rangle = \sum_{i,j \in \mathcal{H}} c_{ij} |i\rangle_1 \otimes |j\rangle_2$$

Extending the same idea of the tensor product to the operators, any Hermitian operator $\mathcal{O}$ may be written as a sum of tensor products of these operators acting on two qubits. To reduce the confusion, consider we apply a one qubit gate G to qubit one, and we don't disturb the second qubit (or use identity operator), and then the operator will be denoted as

$$G_1 = G \otimes \mathbb{I}$$

observe the 1 in subscript of G, this represents that we have applied G on qubit one alone.

Controlled Gates: Controlled gates act on two or more qubits, where one or more qubits act as a control for some operation. We will restrict ourselves to two-qubit controlled gates. Consider a unitary operator U,

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}.$$

And we want to apply this operator on qubit 2 when controlled by qubit 1. This statement can be rephrased as 'if the control qubit is in state $|0\rangle$, then do nothing, but if it is in state $|1\rangle$, then apply the unitary gate U on target qubit.' Say this 'control-U' gate is represented as CU in quantum circuits, and this gate maps the computational basis as follows:

$$|00\rangle \xmapsto{CU} |00\rangle$$
$$|01\rangle \xmapsto{CU} |01\rangle$$
$$|10\rangle \xmapsto{CU} |1\rangle \otimes U |0\rangle$$
$$|11\rangle \xmapsto{CU} |1\rangle \otimes U |1\rangle$$

With this mapping, we can compute the matrix representation of CU as

$$CU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{pmatrix} \tag{2.8}$$

We can have a Hamiltonian representation for this controlled unitary operation as

$$CU = \exp\left(-i\frac{\pi}{4}[\mathbb{I} - Z_1][\mathbb{I} - U_2]\right) \tag{2.9}$$

here, $\mathbb{I}$ represent a 4×4 identity matrix; also take a note about subscripts in Z and U gates.

§2.2.2 Toy example

In this example, we will see how these different gates and concepts can result in good results.

For our toy example, we will evolve our system under 'control-Hadamard gate' (CH). We will do a projective measurement of our system after the unitary transformation.

Recall matrix representation of Hadamard gate from eq. (2.7). Using the time-evolution form, we can find the matrix of the control Hadamard gate (given in eq. (2.9)). We have

$$CH = \exp\left(-i\frac{\pi}{4}[\mathbb{I} - Z_1][\mathbb{I} - H_2]\right)$$

And using `Mathematica`, we can compute the exact form of the matrix (see code B.2)

$$CH = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{2.10}$$

Since measurements are very non-trivial in quantum mechanics, and we don't have any classical analog of the same, it isn't easy to simulate using classical systems. We will use projective measurements (of Z) in our discussion as we can find the matrix for the same (see section 2.3.1). Say we want to project the state of a qubit in Z basis so that we can use the projection operator given by

$$P_0 = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad P_1 = |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

These matrices will be used to calculate the next state after we make a projective measurement for a single qubit. For that, we calculate the corresponding probability as

$$p_0 = \langle\psi|P_0|\psi\rangle$$

So, the state after the measurement will be given by (considering renormalization):

$$|\psi_m\rangle = \frac{P_0|\psi\rangle}{\sqrt{p_0}} \tag{2.11}$$

For our example, say we want to measure the third qubit in $|0\rangle$ state, so the corresponding matrix of projection will be as follows:

$$P_0^{(3)} = \mathbb{I} \otimes \mathbb{I} \otimes P_0 \otimes \mathbb{I}$$

§2.2.2.1 System construction

> **Example 10 (Four Qubit QC)**
> Consider a quantum many-body system composed of four qubits evolving through time under a unitary operator (CH) arranged in a specific way (brickwork structure, see section 2.3). After the evolution, we project the third qubit in the state $|1\rangle$. For a schematic diagram of the quantum circuit of this example, see fig. 3. We prepare our system's initial state as follows:
>
> $$|\Psi\rangle = |+\rangle \otimes |-\rangle \otimes |+\rangle \otimes |-\rangle = |+-+-\rangle$$

Generally, qubits are initialized in the state $|0\rangle$, so we must create the qubits in our desired state using appropriate unitary transformations (see the use of X and H gates in the circuit).
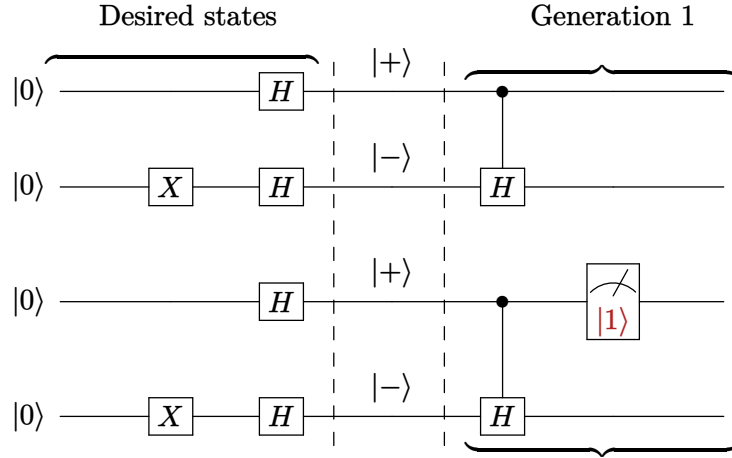
Figure 3: Quantum circuit of toy model

§2.2.2.2 Calculating Final State

Now we know our quantum circuit, so we find the final state using a classical simulator (using Julia see code B.3). Define the unitary transformation matrix as $U_1 = CH \otimes CH$, and we can visualize the state evolution as follows:

$$
|\Psi\rangle = \begin{pmatrix} 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \end{pmatrix} \xrightarrow{U_1} \begin{pmatrix} 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.0i \\ 0.35 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ 0.0i \\ -0.35 + 0.0i \\ 0.0i \\ 0.0i \\ 0.0i \\ 0.0i \\ 0.35 + 0.0i \\ -0.35 + 0.0i \\ 0.0i \\ 0.5 + 0.0i \end{pmatrix} = |\Phi_e\rangle \xrightarrow{P_1^{(3)}} \begin{pmatrix} 0.0i \\ 0.0i \\ 0.0i \\ 0.5 + 0.0i \\ 0.0i \\ 0.0i \\ 0.0i \\ -0.5 + 0.0i \\ 0.0i \\ 0.0i \\ 0.0i \\ 0.0i \\ 0.0i \\ 0.0i \\ 0.0i \\ 0.71 + 0.0i \end{pmatrix} = |\Phi_m\rangle
$$

We observed that we must renormalize the state after the projective measurement (see section 2.3.1). Using the probability of the evolved state $|\Phi_e\rangle$, as

$$
|\Phi_m\rangle = \frac{P_1^{(3)} |\Phi_e\rangle}{\sqrt{p_1}}, \qquad p_1 = \left\langle \Phi_e \middle| P_1^{(3)} \middle| \Phi_e \right\rangle
$$

We can see that we started with an equal superposition state $|\Psi\rangle$, but the final state $|\Phi_m\rangle$ has different properties compared to the starting state:

- We can observe that the probability of finding qubits 3 and 4 in the state $|0\rangle$ is zero,

- probability of finding qubit 1 in the state $|0\rangle$ is 1/2,

- and the probability of finding qubit 2 in the state $|1\rangle$ is 3/4.

§2.3 **Brickwork Structure**

Extending our toy example, we now use a chain of L qubits under unitary dynamics as our quantum many-body system. The unitary evolution will be discrete in time $t \in \mathbb{Z}$. For each time step t, we will be having a unitary evolution operator $U_t = U(t; t-1)$ and say at time $t-1$ system is in state $|\psi_{t-1}\rangle$ then state will evolve as

$$|\psi_t\rangle = U_t |\psi_{t-1}\rangle \tag{2.12}$$

Say we want to compute the final state of the system given the state at time $t = 0$; we do as follows:

$$U(t; 0) = U_t \cdots U_2 U_1 \qquad \Rightarrow \qquad |\psi_t\rangle = U(t; 0) |\psi_0\rangle \tag{2.13}$$

U_t is a tensor product of local unitary operators acting on a pair of qubits.

$$U_\tau = \begin{cases} \displaystyle\bigotimes_{x \in \text{odd bonds}} u_{\tau,x} & \text{if } \tau \text{ is odd,} \\[2em] \displaystyle\bigotimes_{x \in \text{even bonds}} u_{\tau,x} & \text{if } \tau \text{ is even.} \end{cases} \tag{2.14}$$

We label all the pairwise unitary operators by the position of the control qubit as x (referred to as bonds). Here $u_{\tau,x}$ is a local unitary operator acting on a pair of qubits connected by bond x (*i.e.* a linear transformation on 4-dimensional Hilbert space), which is applied in time-step τ.
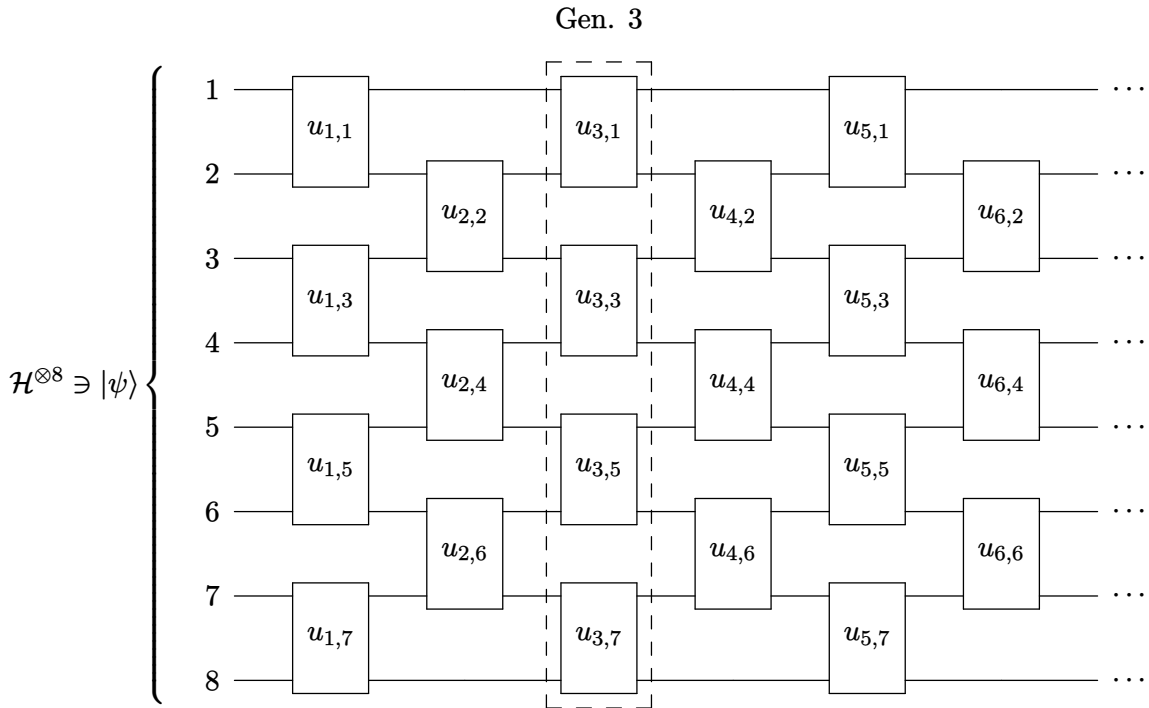

Figure 4: A spacetime diagram of 8 qubits RQC. This arrangement of two-site unitary gates will be called a brickwork structure.

See fig. 4, we have considered a chain of 8 qubits as our quantum many-body system. To become familiar with the notation regarding the unitary transformations, look at the marked generation 3. The unitary operator for $2 \to 3$ transition is given by:

$$U_3 = \bigotimes_{x \in \text{odd bonds}} u_{3,x} = u_{3,1} \otimes u_{3,2} \otimes u_{3,5} \otimes u_{3,7}$$

We can see that the circuit grows from left to right, and the index τ is increasing, implying the direction of time from *i.e.* from left to the right.

If we construct our RQCs in this described fashion (*i.e.* brickwork structure), we can note an underlying property that makes these circuits a good contender for QCAs *i.e.* spatial locality. And if we introduce a fixed pattern on U_τ, we will retrieve time universality.

This circuit has *no conservation laws* (even energy conservation is not conserved). But this system has local equilibrium states. Under the unitary evolution, an arbitrary initial state equilibrates to a local Gibbs state[1].

§2.3.1 Measurements

We know what a measurement means in the context of quantum physics. But how this measurement affects our quantum circuits is something new. Measurements affect the state of a qubit. In RQCs, the state of the neighbouring qubits depends upon the state of the measured qubit. If the number of measurements is significant in a given spacetime volume (*i.e.* in a defined random quantum circuit), we may see altered system dynamics.

Say we measure the ith qubit in a given quantum circuit in computational basis (*i.e.* Z basis). If the state of the quantum circuit before the measurement is given by $|\psi\rangle$, then the state vector after measurement will be

$$|\psi\rangle \rightarrow \begin{cases} \dfrac{P_0^{(i)}|\psi\rangle}{p_0^{(i)}} & \text{with probability } p_0^{(i)} = \left\langle\psi\middle|P_0^{(i)}\middle|\psi\right\rangle \\[2em] \dfrac{P_1^{(i)}|\psi\rangle}{p_1^{(i)}} & \text{with probability } p_1^{(i)} = \left\langle\psi\middle|P_1^{(i)}\middle|\psi\right\rangle \end{cases} \tag{2.15}$$

A local projective measurement disentangles the qubit i from the system.

If an RQC contains projective measurement operators, we call those circuits "`monitored`" quantum circuits. For classical computers, these monitored random quantum circuits work as models for a quantum phase transition in computational complexity.

§2.3.2 Extending Toy example

See example 10 we can continue to apply unitary gates in the same manner (as constructed in section 2.3). Consider using the brickwork structure on that example for four generations.
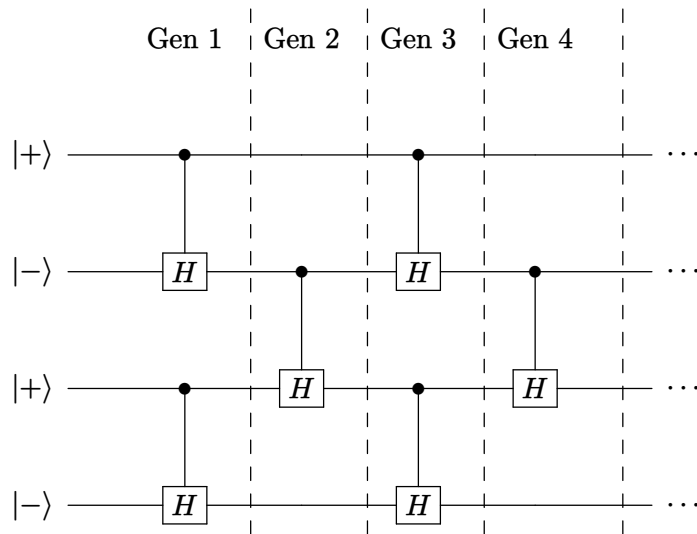

Figure 5: Extended version of our toy model quantum circuit

---

[1] https://en.wikipedia.org/wiki/Gibbs_state

Using the same methodology, we can compute the final state of this quantum system with the Julia program (see code B.4). With the program, we can compute the final state for any generation τ with the function `state_evolution`

$$|\Psi\rangle = \begin{pmatrix} 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \end{pmatrix} \xrightarrow{U(4;0)} \begin{pmatrix} 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ -0.25 + 0.0i \\ 0.0i \\ -0.35 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \\ 0.177 + 0.0i \\ 0.073 + 0.0i \\ -0.177 + 0.0i \\ 0.427 + 0.0i \\ 0.0i \\ -0.35 + 0.0i \\ -0.25 + 0.0i \\ 0.25 + 0.0i \end{pmatrix} = \left|\Phi_e^{(4)}\right\rangle$$

§2.4 Fidelity – a measure of memory

Quantum Cellular Automata are essential for quantum computation as they give rise to exciting physics as we change gates and projective measurements. However, due to the peculiar nature of quantum physics, it is not easy to encode our classical information in quantum computers.

So, if we use quantum computers, we must check the output states (are they as expected or not). For the same, we require a quantity that works as an indicator in this case.

> **Definition 11 (Fidelity):**
> Fidelity measures "overlap" between two states from a given Hilbert space $\mathcal{H}$. Consider two states $|\psi\rangle, |\phi\rangle \in \mathcal{H}$; we define fidelity between these two states as follows:
>
> $$F(\psi, \phi) = |\langle\psi|\phi\rangle|^2 \tag{2.16}$$

Fidelity can be interpreted in many different ways:

<u>Probabilistic Interpretation:</u> It is the probability that the state $|\psi\rangle$ will pass the projective measurement onto the state $|\phi\rangle$.

<u>Metric on Hilbert Space:</u> It is the cosine of angular distance between two states in the given Hilbert space.

We can observe the following properties of this quantity:

- Symmetry: $F(\psi, \phi) = F(\phi, \psi)$.

- Bounded Values: For any ψ, ϕ, we have

$$0 \leq F(\phi, \psi) \leq 1 \quad \text{and} \quad F(\psi, \psi) = 1 = F(\phi, \phi)$$

> Remark 12 (Angle between two states). Building upon the "metric" interpretation, we define *angle between two states* as:
> $$A(\psi, \phi) = \cos^{-1}(F(\psi, \phi)).$$
> The angle is non-negative, symmetric in its inputs, and is equal to zero if and only if $\psi = \phi$. This

> follows triangle inequality, which implies this can be used as a metric for our interested space.

§2.4.1 Use of Fidelity as a measure of memory

Consider we encode information in the initial state for our RQC, and now we want to know how fast (or slow) our data gets lost in the quantum circuit. For the same, we can answer this question with the help of fidelity. Let's prepare an initial state, say

$$|\psi\rangle = |+-+-\rangle$$

And we vary our RQC as follows:

1. For this system, we will use CY gate as our pairwise operator.
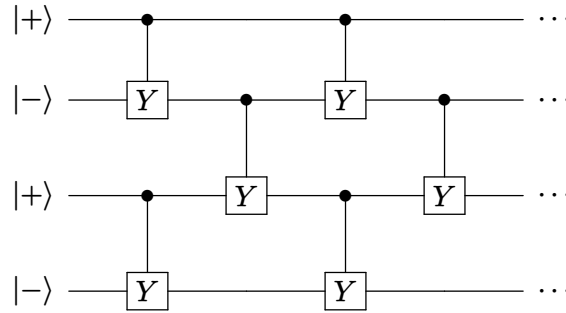
Figure 6: RQC with CY as pairwise operator with initial state $|\psi\rangle$

Now, if we calculate the fidelity of τth generation and initial state. We get

$$F_\psi(\tau) := F(\psi_\tau, \psi) = |\langle \psi_\tau | \psi \rangle|^2 \qquad |\psi_\tau\rangle = U(\tau; 0) |\psi\rangle$$

If plot Fidelity $F_\psi(\tau)$ vs number of generation for $0 \leq \tau \leq 100$, we get the following plot.
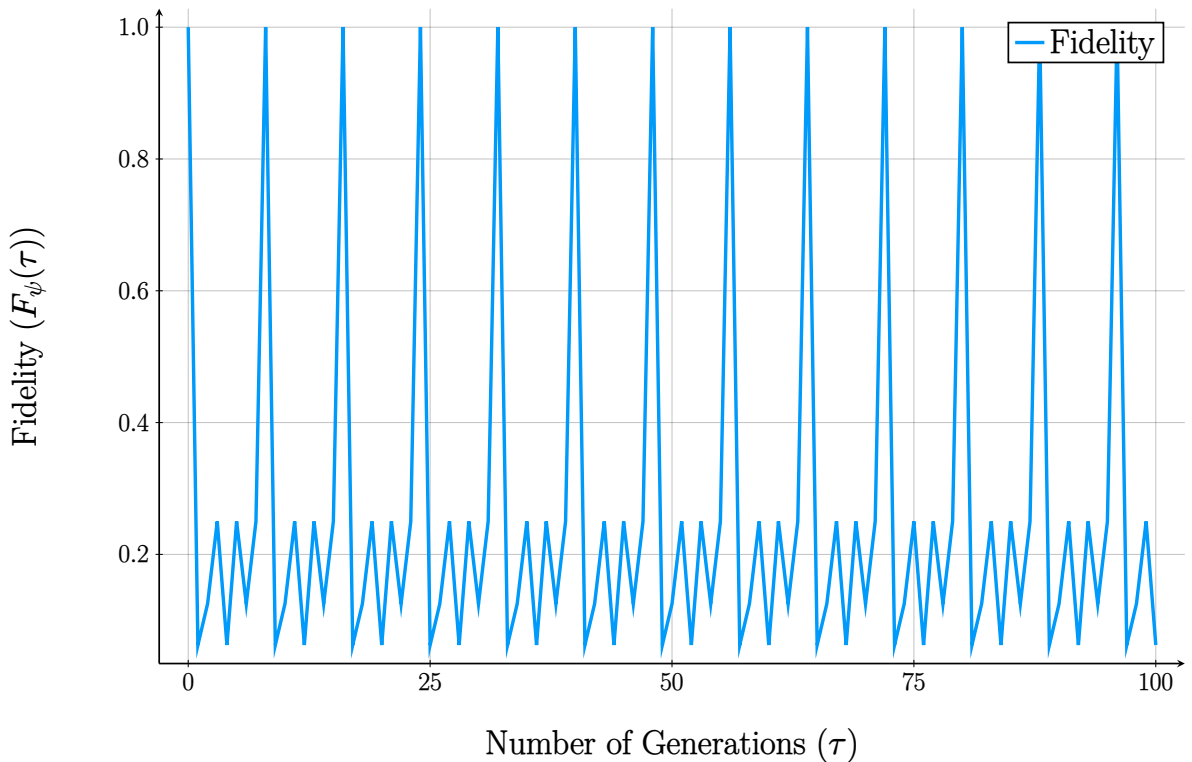
Figure 7: Fidelity $F_\psi(\tau)$ vs Number of generations τ [for CY operator]

2. For this system, we will use CZ gate as our pairwise operator.
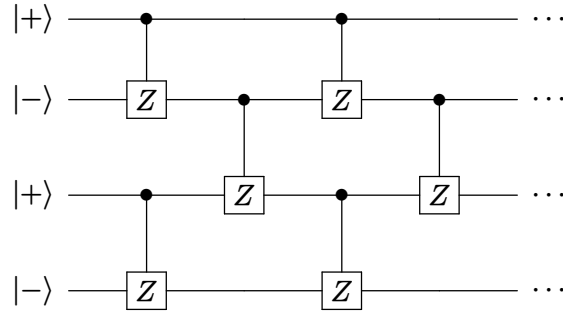
Figure 8: RQC with CZ as pairwise operator with initial state $|\psi\rangle$

With the circuit in fig. 8, we get the following graph if all the conditions are the same as for the CY case (see fig. 6).
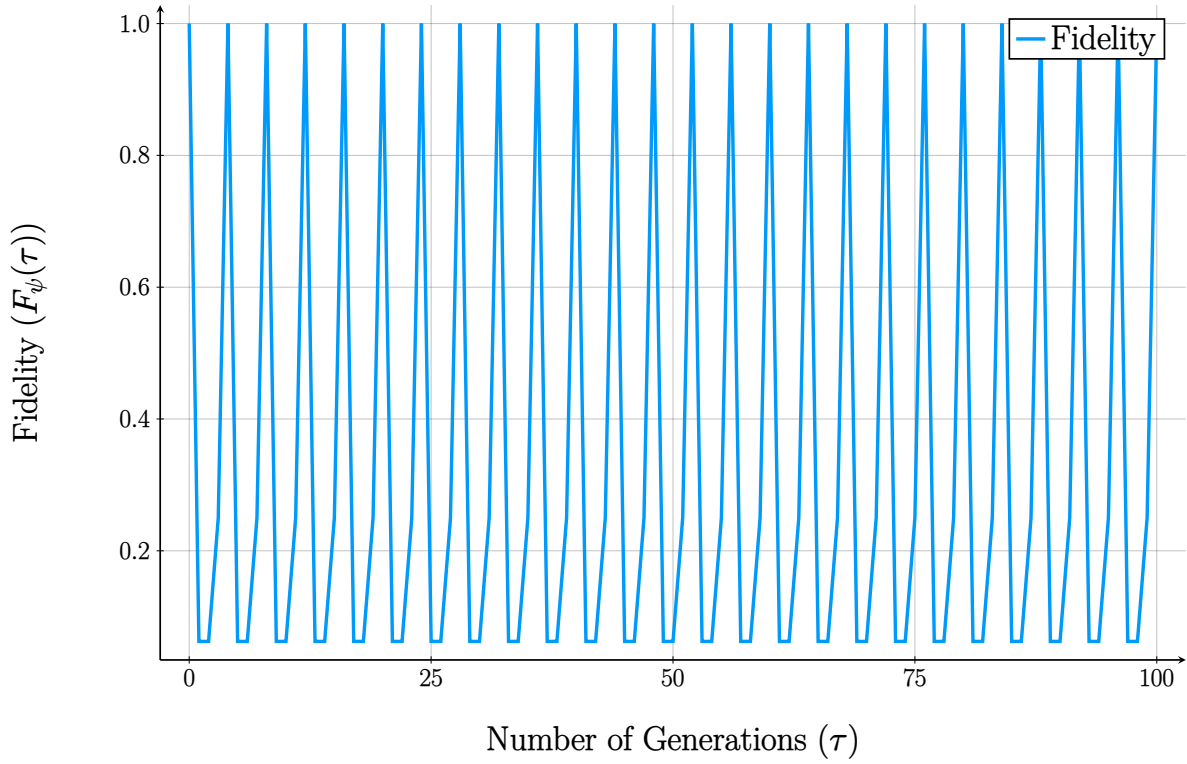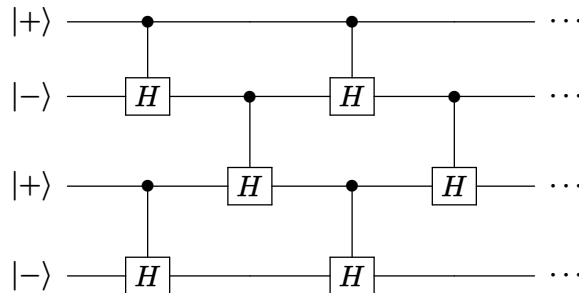

Figure 9: Fidelity $F_\psi(\tau)$ vs Number of generations τ [for CZ operator]

3. For this system, we will use our good old CH gate as our pairwise operator.

Figure 10: RQC with CH as pairwise operator with initial state $|\psi\rangle$

Considering the circuit in fig. 10, we get the following plot for the same,
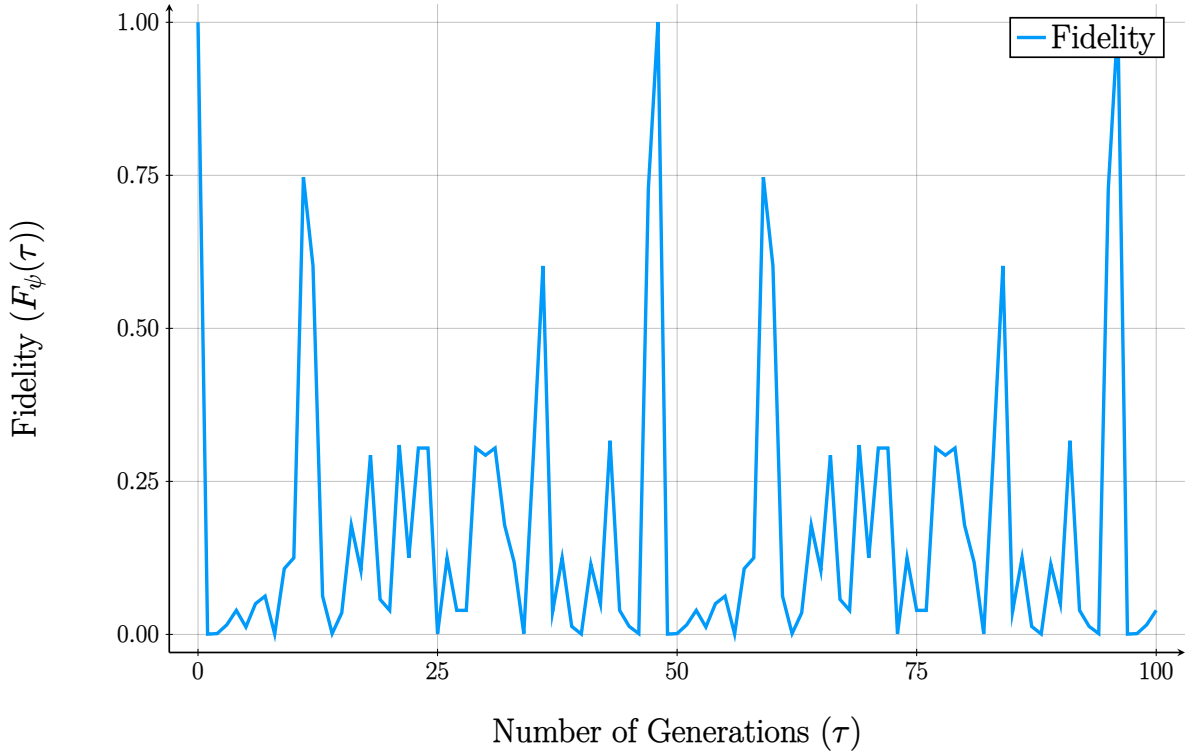
Figure 11: Fidelity $F_\psi(\tau)$ vs Number of generations τ [for CH operator]

§2.4.1.1 Observations & Conclusion

We can observe the following properties of these quantum circuits from these plots:

- There are multiple instances where the fidelity is one other than for $\tau = 0$, which implies the information is not entirely lost in these unitary transformations.

- In complement to that, we have many generations where the fidelity is zero, which means we have entirely lost the information we encoded in the initial state.

- If we look at the graphs, we can see the periodic nature of the fidelity. If we see each graph individually:

  1. See fig. 7, we can observe a periodic behavior of fidelity in the RQC with pairwise operator CY. If we look at the fidelity values, we can determine the period to be 8 generations.

  2. See fig. 9, we can observe a periodic behavior of fidelity in the RQC with pairwise operator CZ. If we look at the fidelity values, we can determine the period to be 4 generations.

  3. See fig. 11, we can observe a periodic behavior of fidelity in the RQC with pairwise operator CH. If we look at the fidelity values, we can determine the period to be 48 generations.

- In the case of CY and CH, we can see there are secondary maxima as well, implying there are states between the full overlap and no overlap fidelities.

We can see these RQCs have a periodic nature for the initial states. Periodicity of fidelity in these circuits can be seen as an emergent property of these minimal-structured (or brickwork-structured) circuits. It also implies that these circuits have a memory of their initial states.

§2.5 Operator Spread – Diffusion of expectation value

In section 2.4.1, we have quantified a physical quantity named "Fidelity", which measures an RQC's memory. Now, let's focus on the problem of how well an RQC transmits information.

Consider we encoded some classical information in the expectation value of an operator for a quantum system. So, we want to find the transmission of the expectation value of that operator through the whole system.

> **Definition 13 (Expectation Value):**
> The expectation value is the probabilistic expected value of a measurement of a quantum state.
> Consider an operator $\mathcal{O}$ defined on a Hilbert space $\mathcal{H}$ corresponding to a quantum system. The expectation value is defined as
>
> $$\langle \mathcal{O} \rangle_\psi = \langle \psi | \mathcal{O} | \psi \rangle \tag{2.17}$$
>
> for any normalized quantum state $|\psi\rangle \in \mathcal{H}$.

Consider a quantum many-body system composed of 8 qubits, evolving through time under the CH gate arranged in brickwork fashion. In this case, our classical information is encoded by operator S_Z.

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

For transmission purposes, say at site 4 (*i.e.* 4$^{\text{th}}$ qubit), the expectation value of this operator is non-zero ($= 1$), and at all other sites, it is zero. We have to construct an initial state that satisfies all these properties, and the easiest state is

$$|\psi\rangle = |+\rangle \otimes |-\rangle \otimes |-\rangle \otimes |0\rangle \otimes |+\rangle \otimes |-\rangle \otimes |+\rangle \otimes |-\rangle$$

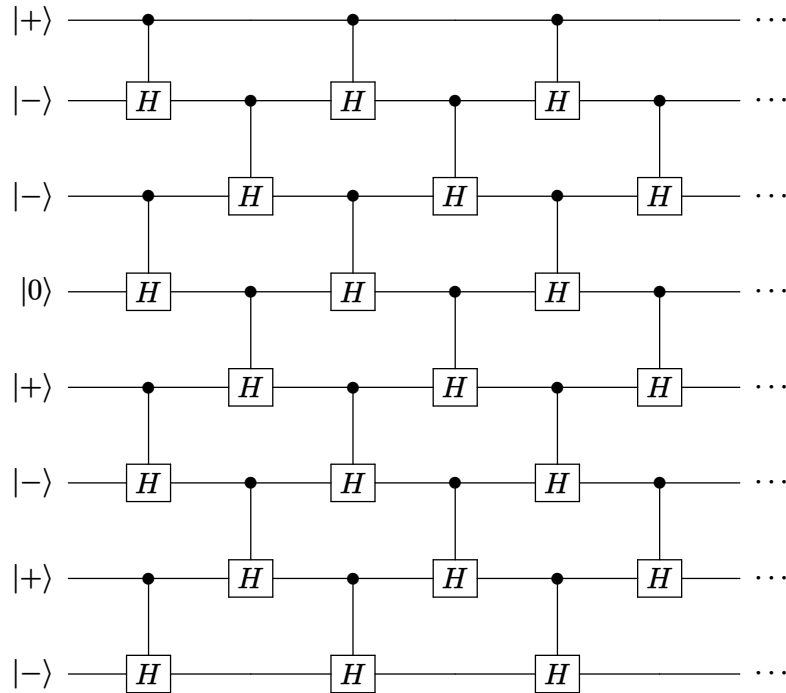Now, if we look at the corresponding quantum circuit schematic

Figure 12: RQC with CH as pairwise operator with initial state $|\psi\rangle$

Let's create a heatmap plot that shows the flow of the expectation value of the S_Z operator with flowing time τ. With the different colors of the heatmap, we can deduce the direction of transmission and even speed.
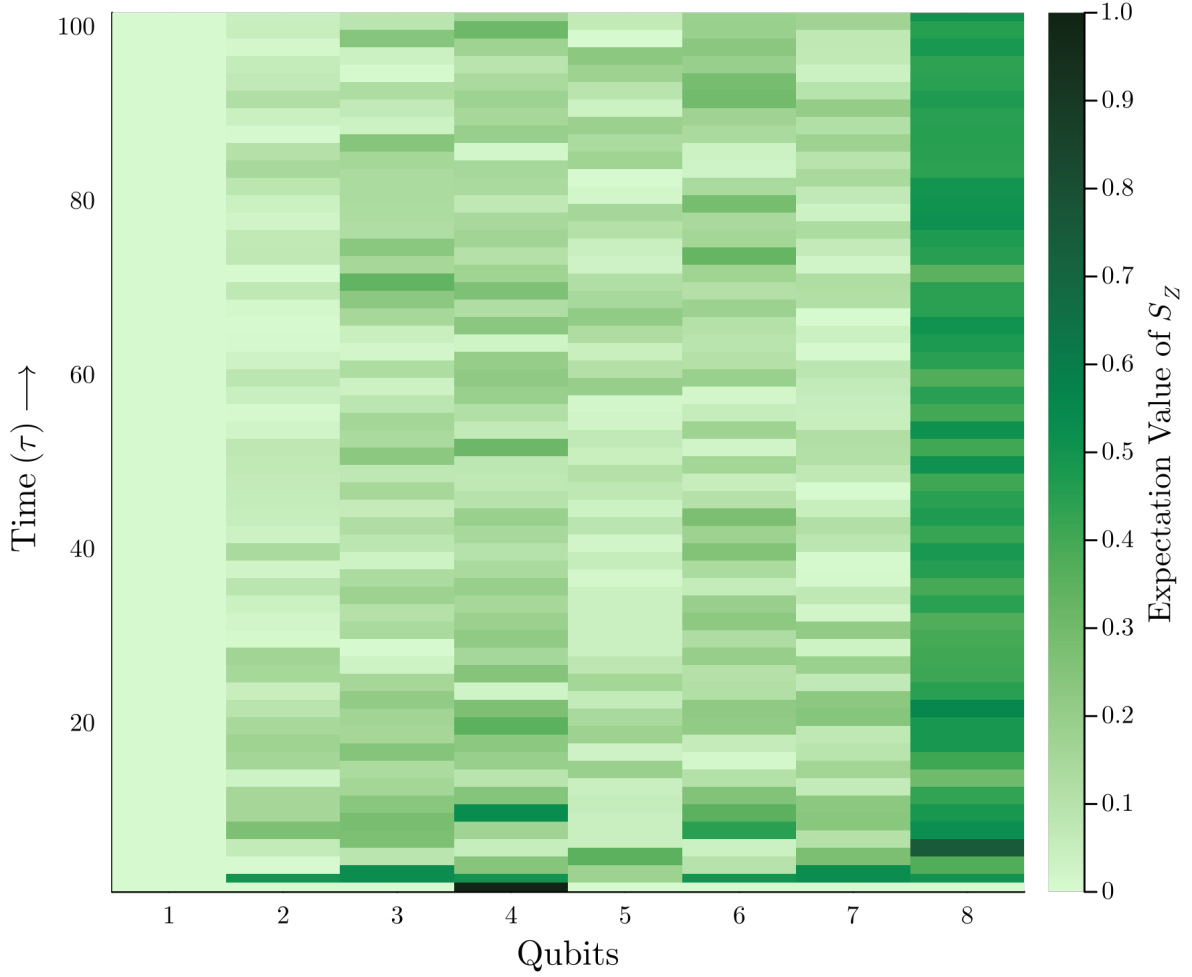

Figure 13: Heatmeat showing the spread of the information which is stored in qubit four at time $\tau = 0$.

§2.5.1 Observation and Conclusion

With the heat map plot, we can deduce the following properties of this system:

1. For qubit one, operator S_Z has an expectation value of zero, which implies there is no gain of information by qubit one.

2. For qubits two to seven, the expectation value of operator S_Z has increased after the first generation. However, there are multiple instances where the value also touches zero, implying these qubits get information but very superficially. Qubits get the information but cannot store it for long.

3. For qubit eight, we see a continuous band of dark green color, implying a very high expectation value of operator S_Z, indicating that information has traveled from qubit four to eight and gets stored there.

From these properties, we can imply that the information encoded in qubit four has traveled to eight, spreading information in bits and pieces to qubits two to seven, suggesting that the spread is constant and then it just stays there.

A Quantum Computers

§A.1 Quantum Gates

In section 2.2.1, we have gone through a few quantum gates. Now, we will look at all the quantum logic gates.
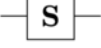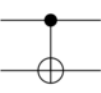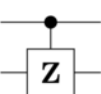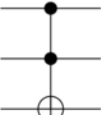
Operator	Gate(s)	Matrix
Pauli-X (X)	—[X]— ⊕	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	—[Y]—	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	—[Z]—	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	—[H]—	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	—[S]—	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)	—[T]—	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Figure 14: Table of all quantum logic gates[1]

---

[1] https://en.wikipedia.org/wiki/Quantum_logic_gate

B Code

This chapter contains all the codes used to generate the graphs or numerical results.

§B.1 Cellular Automata

This section contains the code used to generate the heatmap diagram for CA named rule-110.

```julia
1   using Plots;
2
3   rule = Dict([0, 0, 0] => 0, [0, 0, 1] => 1, [0, 1, 0] => 1, [0, 1, 1
        ] => 1, [1, 0, 0] => 0, [1, 0, 1] => 1, [1, 1, 0] => 1, [1, 1, 1]
        => 0)
4
5   function next_generation(state)
6       new_state = zeros(Int, length(state))
7       for i in 2:length(state)-1
8           neighbour = state[i-1:i+1]
9           new_state[i] = rule[neighbour]
10      end
11      return new_state
12  end
13
14  function generate_automaton(generations, initial_state)
15      automaton = zeros(Int, generations, length(initial_state))
16      automaton[1, :] = initial_state
17      for i in 2:generations
18          automaton[i, :] = next_generation(automaton[i-1, :])
19      end
20      return automaton
21  end
22
23  initial_state = zeros(Int, 101)
24  initial_state[100] = 1
25
26  # Generate the Rule 110 cellular automaton with 100 time steps
27  generations = 100
28  automaton = generate_automaton(generations, initial_state);
29
30  # Plot the cellular automaton
31  p = heatmap(automaton, c=:cividis, xlabel="Cell Index", ylabel="Time
        Steps", cbar=false)
32  plot!(
33      # size=(0, 550),
34      # guidefontsize=20,
35      # tickfontsize=15,
36      dpi=300,
37      background_color=:transparent,
38      fontfamily="Computer Modern"
39  )
40  savefig(p, "rule110_cellular_automaton.png")
```

```
41  display(p)
```

Code B.1: Generating Spacetime diagram of CA `rule 110`

§B.2 Toy example

```
1   Z = PauliMatrix[3];
2   H = HadamardMatrix[2];
3   Id = IdentityMatrix[4];
4   Z1 = KroneckerProduct[Z, IdentityMatrix[2]];
5   H2 = KroneckerProduct[IdentityMatrix[2], H];
6   CH = Simplify[MatrixExp[- I \[Pi]/4 (Id-Z1).(Id-H2)]];
7   MatrixForm[CH]
```

Code B.2: Computing matrix of CH gate

```
1   using LinearAlgebra, QuantumInformation, Latexify
2
3   # Impotant Const
4   u, d = ket(1, 2), ket(2, 2);
5   r, l = 1 / sqrt(2) * (u + d), 1 / sqrt(2) * (u - d);
6
7   Id, X, Y, Z, H = Matrix{Int}(I, 2, 2), [0 1; 1 0], [0 -im; im 0], [1
        0; 0 -1], 1 / sqrt(2) * [1 1; 1 -1];
8
9   CH = [
10      1 0 0 0;
11      0 1 0 0;
12      0 0 1/sqrt(2) 1/sqrt(2);
13      0 0 1/sqrt(2) -1/sqrt(2)
14  ]
15  Pd = [0 0; 0 1]
16
17  # Initial State of the system as |+-+-⟩
18  ψ = kron(r, l, r, l)
19
20  ## Generation 1 ##
21
22  # Unitary Operators #
23  U1 = kron(CH, CH);
24  φ1e = U1 * ψ
25
26  # Projective Measurement #
27  Pd3 = kron(Id, Id, Pd, Id);
28  p1 = real(φ1e' * Pd3 * φ1e)
29  φ1m = φ1m = Pd3 * φ1e / sqrt(p1)
30  print(latexify(φ1m))
```

Code B.3: Simulating the final state of toy example using matrix operations

```julia
1  using LinearAlgebra, QuantumInformation, Latexify
2
3  u, d = ket(1, 2), ket(2, 2);
4  r, l = 1 / sqrt(2) * (u + d), 1 / sqrt(2) * (u - d);
5
6  Id, X, Y, Z, H = Matrix{Int}(I, 2, 2), [0 1; 1 0], [0 -im; im 0], [1
       0; 0 -1], 1 / sqrt(2) * [1 1; 1 -1];
7
8  CH = [1 0 0 0; 0 1 0 0; 0 0 1/sqrt(2) 1/sqrt(2); 0 0 1/sqrt(2) -1/
       sqrt(2)]
9
10 ψ = kron(r, l, r, l)
11
12 function state_evolution(state::Vector{ComplexF64},
       no_of_generations)
13     odd_gen_mat = kron(CH, CH)
14     even_gen_mat = kron(Id, CH, Id)
15     counter = 0
16     final_state = state
17     while counter < no_of_generations
18         counter += 1
19         if counter % 2 == 1
20             final_state = odd_gen_mat * final_state
21         else
22             final_state = even_gen_mat * final_state
23         end
24     end
25     return final_state
26 end
27
28 ψ_F = state_evolution(ψ, 4)
29 print(latexify(ψ_F))
```

Code B.4: Simulating the final state of our extended system using matrix operations

§B.3 Fidelity Calculations

```julia
1  using Plots, LinearAlgebra, QuantumInformation, LaTeXStrings;
2  pgfplotsx();
3
4  u, d = ket(1, 2), ket(2, 2);
5  r, l = 1 / sqrt(2) * (u + d), 1 / sqrt(2) * (u - d);
6
7  Id, X, Y, Z, H = Matrix{Int}(I, 2, 2), [0 1; 1 0], [0 -im; im 0], [1
       0; 0 -1], 1 / sqrt(2) * [1 1; 1 -1];
8
9  CY = [1 0 0 0; 0 1 0 0; 0 0 0 -1im; 0 0 1im 0];
10 CZ = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 -1];
```

```julia
11  CH = [1 0 0 0; 0 1 0 0; 0 0 1/sqrt(2) 1/sqrt(2); 0 0 1/sqrt(2) -1/
        sqrt(2)];
12  ψ = kron(r, l, r, l)
13
14  function state_evolution(state::Vector{ComplexF64},
        no_of_generations)
15      # odd_gen_mat = kron(CY, CY)
16      # even_gen_mat = kron(Id, CY, Id)
17      odd_gen_mat = kron(CZ, CZ)
18      even_gen_mat = kron(Id, CZ, Id)
19      # odd_gen_mat = kron(CH, CH)
20      # even_gen_mat = kron(Id, CH, Id)
21      counter = 0
22      final_state = state
23      while counter < no_of_generations
24          counter += 1
25          if counter % 2 == 1
26              final_state = odd_gen_mat * final_state
27          else
28              final_state = even_gen_mat * final_state
29          end
30      end
31      return final_state
32  end
33
34  function fidelity(initial_state::Vector{ComplexF64}, final_state::
        Vector{ComplexF64})
35      return abs(initial_state' * final_state)^2
36  end
37
38  time = 0:1:100;
39
40  fidelity_plot = [fidelity(state_evolution(ψ, t), ψ) for t in time]
41
42  plot_height, plot_width = 11.69, 8.27
43  p = plot(time, fidelity_plot, label="Fidelity", lw=2)
44  plot!(
45      size=(900, 600),
46      extra_kwargs=Dict(:plot => Dict("height" => "$(plot_height)in",
        "width" => "$(plot_width)in")),
47      dpi=300,
48      xlabel=L"Number of Generations $(\tau)$",
49      ylabel=L"Fidelity $(F_{\psi}(\tau))$",
50      legendfontsize=20,
51      titlefontsize=20,
52      tickfontsize=15,
53      guidefontsize=20,
54      ga=0.25,
55      draw_arrow=true,
56      legend=:topright
57  )
58  # savefig(p, "fidelity_plot_CY.tikz")
59  savefig(p, "fidelity_plot_CZ.tikz")
```

```
60  # savefig(p, "fidelity_plot_CH.tikz")
61  display(p)
```

Code B.5: Plotting fidelity vs increasing generations

§B.4 **Operator Spread**

```
1   using Plots, LinearAlgebra, QuantumInformation, LaTeXStrings
2
3   u, d = ket(1, 2), ket(2, 2);
4   r, l = 1 / sqrt(2) * (u + d), 1 / sqrt(2) * (u - d);
5
6   Id, X, Y, Z, H = Matrix{Int}(I, 2, 2), [0 1; 1 0], [0 -im; im 0], [1
        0; 0 -1], 1 / sqrt(2) * [1 1; 1 -1];
7
8   CH = [1 0 0 0; 0 1 0 0; 0 0 1/sqrt(2) 1/sqrt(2); 0 0 1/sqrt(2) -1/
        sqrt(2)]
9
10  Sz = [1 0; 0 -1] # Pauli Matrix Z
11
12  ψ = kron(r, l, l, u, r, l, r, l); # Initial state
13
14  function state_evolution(state::Vector{ComplexF64},
        no_of_generations::Int64)
15      odd_gen_mat = kron(CH, CH, CH, CH)
16      even_gen_mat = kron(Id, CH, CH, CH, Id)
17      counter = 0
18      final_state = state
19      while counter < no_of_generations
20          counter += 1
21          if counter % 2 == 1
22              final_state = odd_gen_mat * final_state
23          else
24              final_state = even_gen_mat * final_state
25          end
26      end
27      return final_state
28  end
29
30  function expectation_value(state::Vector{ComplexF64}, operator::
        Matrix{Int64})
31      return abs(state' * operator * state) < 1e-10 ? 0 : abs(state' *
        operator * state)
32  end
33
34  Sz1 = kron(Sz, Id, Id, Id, Id, Id, Id, Id)
35  Sz2 = kron(Id, Sz, Id, Id, Id, Id, Id, Id)
36  Sz3 = kron(Id, Id, Sz, Id, Id, Id, Id, Id)
37  Sz4 = kron(Id, Id, Id, Sz, Id, Id, Id, Id)
38  Sz5 = kron(Id, Id, Id, Id, Sz, Id, Id, Id)
```

```julia
39  Sz6 = kron(Id, Id, Id, Id, Id, Sz, Id, Id)
40  Sz7 = kron(Id, Id, Id, Id, Id, Id, Sz, Id)
41  Sz8 = kron(Id, Id, Id, Id, Id, Id, Id, Sz)
42
43  operator_list = [Sz1, Sz2, Sz3, Sz4, Sz5, Sz6, Sz7, Sz8]
44
45  function expectation_value_chain(state::Vector{ComplexF64},
        list_of_operators::Vector{Matrix{Int64}})
46       no_of_qubits = Int(log2(length(state)))
47       expectation_cells = zeros(Float64, no_of_qubits)
48       for i in 1:no_of_qubits
49           expectation_cells[i] = abs(expectation_value(state,
        list_of_operators[i]))
50       end
51       return expectation_cells
52  end
53
54  function generate_matrix(state::Vector{ComplexF64}, generation::
        Int64, list_of_operators::Vector{Matrix{Int64}})
55       no_of_qubits = Int(log2(length(state)))
56       matrix_expect = zeros(Float64, generation + 1, no_of_qubits)
57       matrix_expect[1, :] = expectation_value_chain(state,
        list_of_operators)
58       for i in 2:generation+1
59           gen_state = state_evolution(state, i)
60           matrix_expect[i, :] = expectation_value_chain(gen_state,
        list_of_operators)
61       end
62       return matrix_expect
63  end
64
65  operator_spread = generate_matrix(ψ, 100, operator_list)
66
67  h = heatmap(operator_spread, size=(600, 500), dpi=300,
68       # guidefontsize=15, tickfontsize=10,
69       xticks=0:1:8, xlabel="Qubits", ylabel=L"Time $(\tau) \
        longrightarrow$", cbar_title=L"Expectation Value of $S_Z$",
70       c=:inferno, background_color=:transparent, fontfamily="Computer
        Modern")
71  savefig(h, "spread-Sz.png")
72  display(h)
```

Code B.6: Spread of expectation value of operator Z

Bibliography

[1] A. Einstein, B. Podolsky, and N. Rosen. "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?" In: *Physical Review* 47.10 (May 1935), pp. 777–780. ISSN: 0031-899X. DOI: 10.1103/physrev.47.777. URL: http://dx.doi.org/10.1103/PhysRev.47.777.

[2] Terry Farrelly. "A review of Quantum Cellular Automata". In: (Apr. 2019). Quantum 4, 368 (2020). DOI: 10.22331/q-2020-11-30-368. arXiv: 1904.13318v2 [quant-ph]. URL: http://arxiv.org/abs/1904.13318v2.

[3] Richard P. Feynman. "Simulating physics with computers". In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/bf02650179. URL: http://dx.doi.org/10.1007/BF02650179.

[4] Richard P. Feynman. "Quantum mechanical computers". In: *Foundations of Physics* 16.6 (June 1986), pp. 507–531. ISSN: 1572-9516. DOI: 10.1007/bf01886518. URL: http://dx.doi.org/10.1007/BF01886518.

[5] Matthew P.A. Fisher et al. "Random Quantum Circuits". In: *Annual Review of Condensed Matter Physics* 14.1 (Mar. 2023), pp. 335–379. ISSN: 1947-5462. DOI: 10.1146/annurev-conmatphys-031720-030658. arXiv: 2207.14280v1 [quant-ph]. URL: http://dx.doi.org/10.1146/annurev-conmatphys-031720-030658.

[6] Martin Gardner. "Mathematical Games". In: *Scientific American* 223.4 (Oct. 1970), pp. 120–123. ISSN: 0036-8733. DOI: 10.1038/scientificamerican1070-120. URL: http://dx.doi.org/10.1038/scientificamerican1070-120.

[7] John von Neumann. "Theory of Self-Reproducing Automata". In: (1966). URL: https://cba.mit.edu/events/03.11.ASE/docs/VonNeumann.pdf.

[8] B. Schumacher and R. F. Werner. "Reversible quantum cellular automata". In: (May 2004). arXiv: quant-ph/0405174v1 [quant-ph]. URL: http://arxiv.org/abs/quant-ph/0405174v1.

[9] John Von Neumann. *Mathematical foundations of quantum mechanics*. en. Ed. by Nicholas A Wheeler. Princeton, NJ: Princeton University Press, Feb. 2018.

[10] J. Watrous. "On one-dimensional quantum cellular automata". In: *Proceedings of IEEE 36th Annual Foundations of Computer Science*. 1995, pp. 528–537. DOI: 10.1109/SFCS.1995.492583.