Quantum Simulation of Physical Systems (QSPS)

Table of Contents

Introduction

This project explores quantum simulation through classical computing frameworks, focusing on modelling physical systems using quantum principles. It stems from my long-standing interest in quantum mechanics and provides a structured opportunity to apply theoretical knowledge practically. The objective is to simulate the evolution of qubit states using mathematical models, computational environments, and graphical visualizations, ultimately deepening my understanding of foundational quantum concepts.

The following report consists of:

Chapter 1: establishes the mathematical framework—including vectors, complex numbers, and Hilbert spaces—required for understanding quantum systems.

Chapter 2: develops the quantum mechanical principles, such as qubits, superposition, Hamiltonians, and time evolution, built on those mathematical tools.

Chapter 3: maps these concepts to a computational environment, explaining the libraries, objects, and functions used in the simulation.

Chapter 4: presents the program development, showcasing simulation results and analyzing quantum state evolution through visual and numerical methods.

1 Chapter 1: Mathematical Foundations

Quantum mechanics governs how subatomic particles behave, making it essential for understanding quantum computing. To grasp the differences from classical computation, we must first understand the underlying mathematics. Think of a chess set: before discussing the pieces (quantum systems), their moves (quantum mechanics), or strategies (simulations), we must first define the board—the environment in which quantum systems operate.

1.1 Vectors

To begin with, we must understand the linear algebra of vectors. We begin with a set, which is a collection of distinct elements. A field (represented as $\mathbb{F}$) is a set of elements and must satisfy the following axioms (assume all a, b, c $\in \mathbb{F}$).

1.1.1 Field Axioms

Addition

1. Closure: if a, b $\in \mathbb{F}$, then a + b $\in \mathbb{F}$
2. Associativity: (a + b) + c = a + (b + c)
3. Commutativity: a + b = b + a
4. Additive identity: there exists 0 $\in \mathbb{F}$ such that a + 0 = a
5. Additive inverse: For every a $\in \mathbb{F}$, there exists -a $\in \mathbb{F}$ such that a + (-a) = 0

Multiplication Axioms

1. Closure: ab $\in \mathbb{F}$
2. Associativity: (ab)c = a(bc)
3. Commutativity: ab = ba
4. Multiplicative identity: there exists 1 $\in \mathbb{F}$, 1 $\neq$ 0, such that 1a = a
5. Multiplicative inverse: For every a $\neq$ 0, there exists $a^{-1} \in \mathbb{F}$ such that $aa^{-1} = 1$

Distributive Axioms

a(b + c) = ab + ac

1.1.2 Vector Spaces

Popular infinite fields commonly used are $\mathbb{R}$ (the set of real numbers) and $\mathbb{C}$ (the set of complex numbers). An example of a finite field is $\mathbb{F}_2 = (0,1)$.

A vector space $\mathcal{V}$ over a field $\mathbb{F}$ is defined by the following components:

1. A set of vectors $\mathcal{V}$
2. A field of scalars $\mathbb{F}$

3. Two operations:
    - Vector addition: $\mathcal{V}\mathcal{V} \to \mathcal{V}$
    - Scalar multiplication: $\mathbb{F}\mathcal{V} \to \mathcal{V}$

($\vec{v} = [x_1, x_2, x_3]$). $\mathbb{C}^2$ instead of $\mathbb{R}^2$.

1.1.2.1 Vector Space axioms

These operations must satisfy the following **vector space axioms** (for all $u, v, w \in \mathcal{V}$ and all scalars $a, b \in \mathbb{F}$):

- Additive associativity: $(u + v) + w = u + (v + w)$
- Additive commutativity: $u + v = v + u$
- Additive identity: There exists a zero vector 0 such that $v + 0 = v$
- Additive inverse: For each v, there exists $-v$ such that $v + (-v) = 0$
- Multiplicative identity: $1v = v$, where 1 is the multiplicative identity in $\mathbb{F}$
- Distributivity over field addition: $(a + b)v = av + bv$
- Distributivity over vector addition: $a(u + v) = au + av$
- Compatibility of scalar multiplication: $a(bv) = (ab)v$

Thus, the term "vector space $\mathcal{V}$ over field $\mathbb{F}$" is used because the scalars in the operations come from the field $\mathbb{F}$.

1.1.2.2 Example of a vector space over $\mathbb{R}$:

Let us take $\mathbb{R}^2$, (the power represents the number of dimensions this vector space is in) where…

$$\mathbb{R}^2 = \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \mid x, y \in \mathbb{R} \right\}$$

This vector space exists in 2 dimensions and satisfices all required axioms, including the Vector Addition and Scalar Multiplication:

- Vector Addition (x_1, x_2, y_1, $y_2 \in \mathbb{R}^2$) $$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix}$$

- Scalar Multiplication (a $\in \mathbb{R}$ and x, y $\in \mathbb{R}^2$) $$a \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ ay \end{bmatrix}$$

Vectors can be represented as column or row form (e.g., $\vec{v} = [x_1, x_2, x_3]$, with the arrow denoting a vector. While we've considered a 2D real vector space, quantum systems operate in complex vector spaces, so we now shift from $\mathbb{R}^2$ to $\mathbb{C}^2$

1.1.3 Complex Numbers and Imaginary Units

An imaginary unit (**I**) = $\sqrt{-1}$, and a complex number (**z**) = a + b**I** where a,b ∈ $\mathbb{R}$ and **I**= $\sqrt{-1}$ so therefore **z** ∈ $\mathbb{C}$. if b = 0 then the imaginary component of **z** becomes null which leads to z = a ∈ $\mathbb{R}$. It is important to note that all real numbers are a special case of complex numbers, with just the imaginary component (b**I**) = 0.

1.1.3.1 Complex Conjugate

- If a complex number (**z**) = a + b**i** where a, b ∈ $\mathbb{R}$
- Its complex conjugate ($\bar{\mathbf{z}}$)= a – b**i** (the sign of the imaginary component is flipped)

Now we have defined a complex vector space (such as $\mathbb{C}^2$), we know the elements it withholds and the axioms it must abide too. However, this is still just an algebraic structure. A mathematical construct where we can linearly combine elements. The next step is to equip the space with geometric abilities.

1.1.4 Inner Product

The space still lacks the notion of length, angles, distance and geometry so it cannot be used to represent the movement or physical behaviour of subatomic particles. For this, we must make sure the complex vector space has an **Inner product** and is **complete (1.1.5).**

1.1.4.1 Inner product axioms

v,u,z ∈ $\mathbb{C}^2$ (the vector space we are checking the inner product of) and a ∈ $\mathbb{C}$ (an element of the field over which the vector space is defined). Using the equation below , we can determine if the complex vector space is a viable inner product space.

$$\langle u|v\rangle = \sum_n (u_n \bar{v}_n)$$

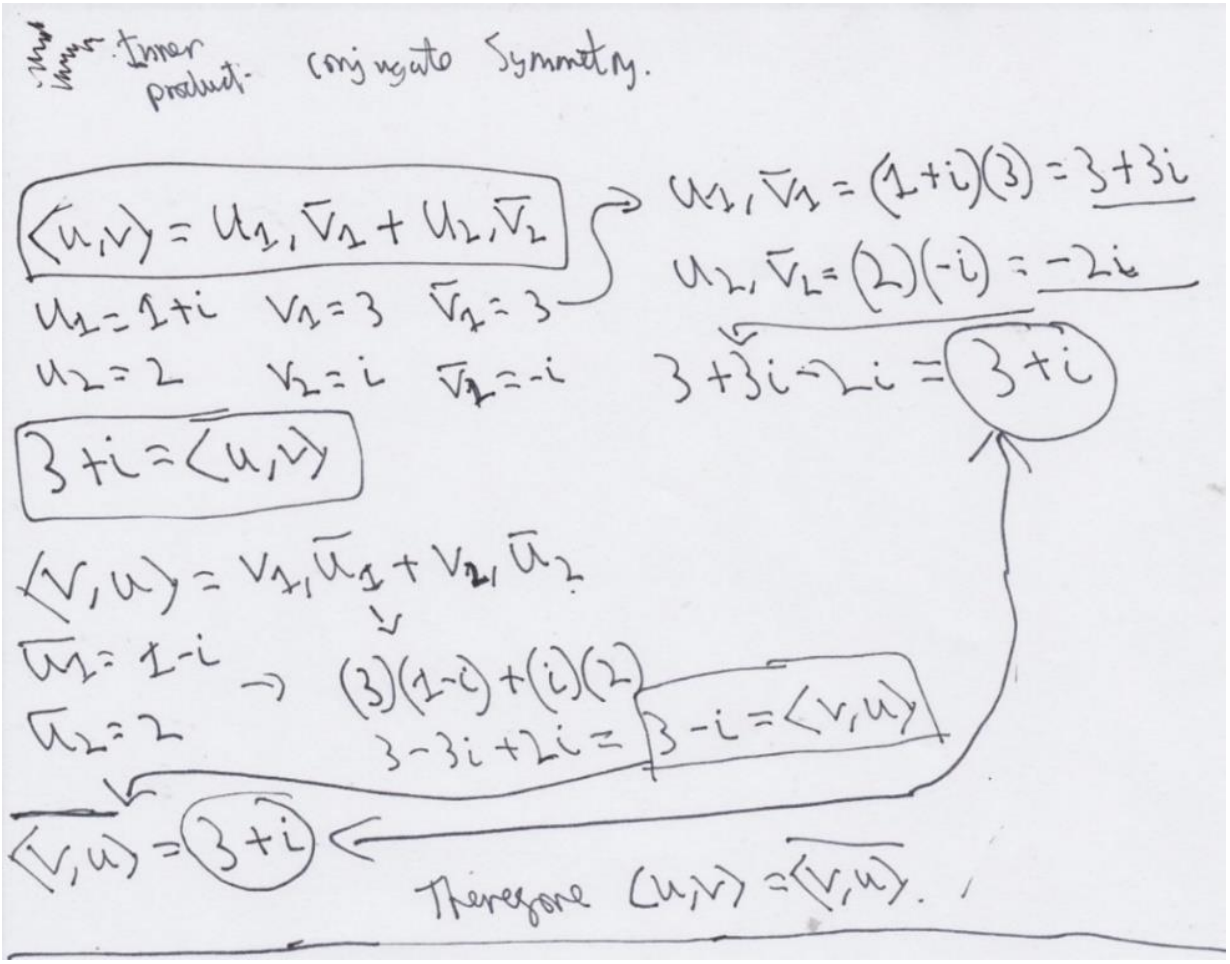
Standard Inner Product Equation

- **Conjugate symmetric:** ⟨ v | u ⟩ = $\overline{\langle u|v\rangle}$
- **Linear in the first argument:** ⟨av + z | u⟩ = a⟨ v | u ⟩ + ⟨ z | u ⟩
- **Positive definite:** ⟨ v | v ⟩ ≥ 0. Only equals 0 if v = 0

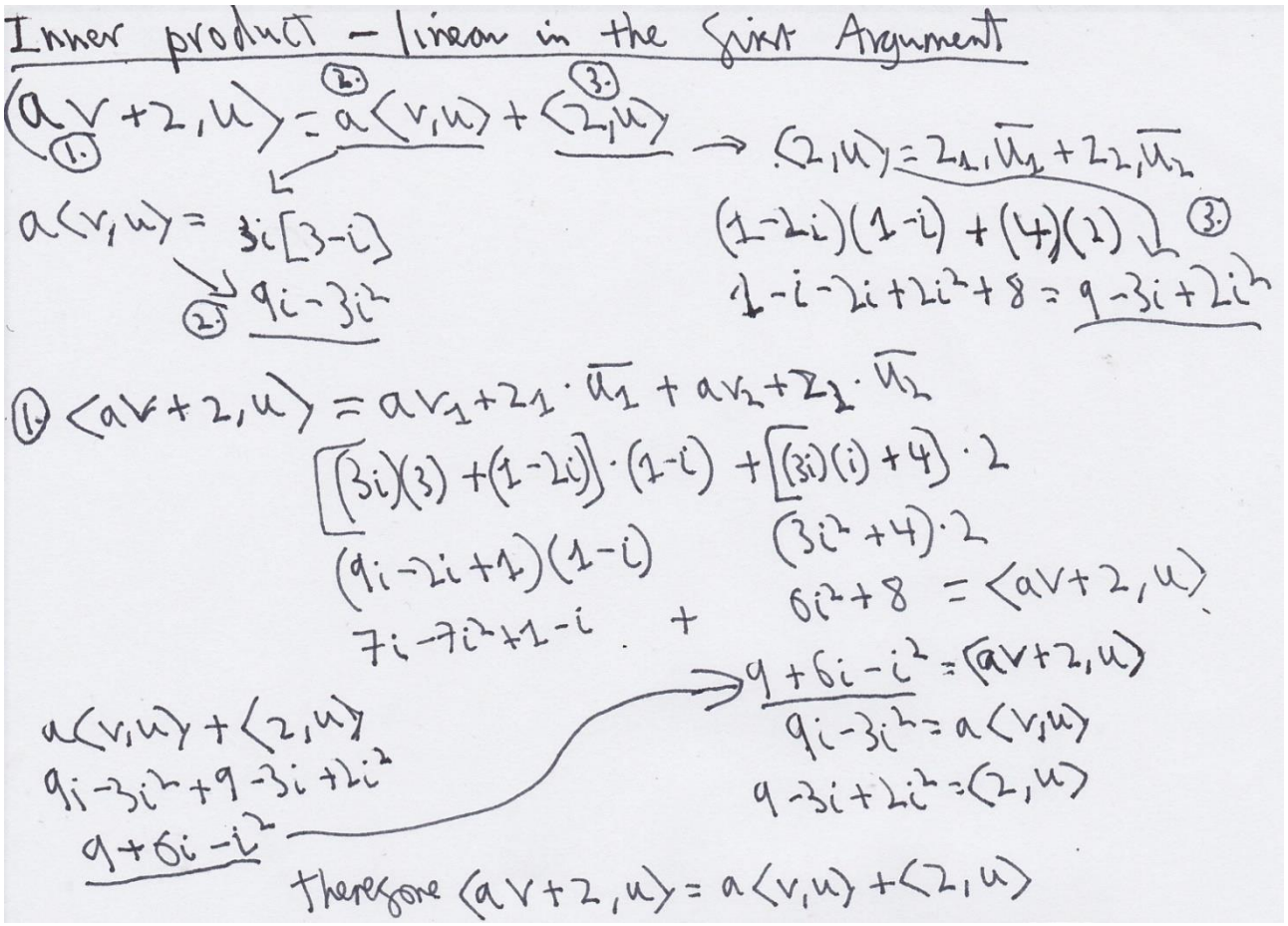
(all Dirac notation, (⟨ | and | ⟩) is explained in more detail at (2.1.1))

Let us see if the inner product of $\mathbb{C}^2$ meets the three requirements above, using the three vectors and scalar $\rightarrow v = \begin{bmatrix} 3 \\ \mathbf{i} \end{bmatrix}, u = \begin{bmatrix} 1+\mathbf{i} \\ 2 \end{bmatrix}, z = \begin{bmatrix} 1-2\mathbf{i} \\ 4 \end{bmatrix}, a = 3\mathbf{i}$

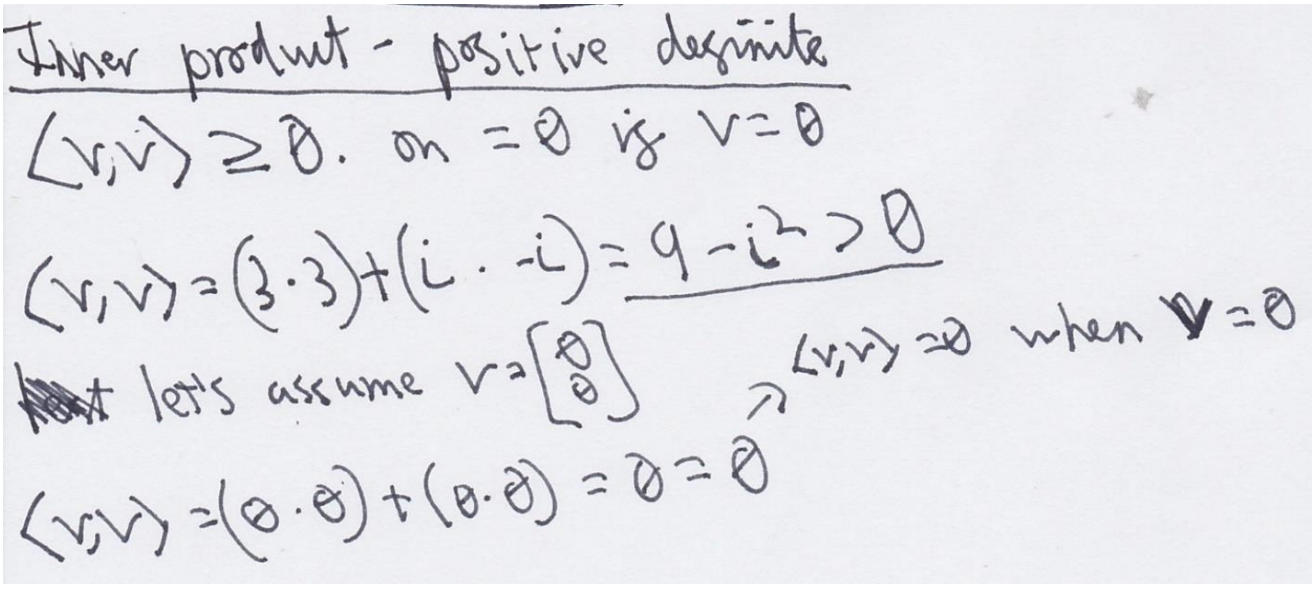
1. Conjugate symmetry:



2. Linear in the first argument:



3. Positive definite:



We have verified that the complex vector space $\mathbb{C}^2$ has a valid inner product by satisfying these three key properties. This provides the following geometric notions in the vector space:

Length	$\|v\| = \sqrt{\langle v, v \rangle}$
Distance	$\|v - w\|$
Angle	$\cos\theta = \frac{\langle v, w \rangle}{\|v\|\|w\|}$
Orthogonality	$\langle v, w \rangle = 0$
Projection	$\text{proj}_u(v) = \langle v, u \rangle u$

1.1.4.2 Outer products

Unlike the inner product where it produces a scalar, an outer product produces a matrix (1.2).

$$\vec{u} \otimes \vec{v} = \vec{u}\vec{v}^{T}$$

Both $\vec{u}$ and $\vec{v}$ must be a one-dimensional matrix (a row or column vector). $\vec{v}^{T}$ = Transpose of $\vec{v}$ (like the complex conjugate of 1.1.4 Inner product), where if :

$$\vec{v} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \rightarrow \vec{v}^{T} = [3 \quad 4 \quad 5].$$

Therefore, if $\vec{u} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \rightarrow \vec{u}\,\vec{v}^{T} = \begin{bmatrix} 1*3 & 1*4 & 1*5 \\ 2*3 & 2*4 & 2*5 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 5 \\ 6 & 8 & 10 \end{bmatrix}$

If dimensions of $\vec{u} = n * 1$ and $\vec{v}^{T} = 1 * m$ then the resulting matrix has dimensions $n * m$. Dirac's notation of the outer product is $|\vec{u}\rangle\langle\vec{v}|$. The result of an outer product is also known as a density matrix but only if the vectors are normalised (2.1.2.1) .

1.1.5 Completeness

$\mathbb{C}^2$ is a finite-dimensional inner product space over the complex field, and as is universally established, all such spaces are complete 3. (Axler, 1997)

1.1.6 Hilbert spaces

Now we have verified that $\mathbb{C}^2$ is a complex vector space equipped with an inner product space and is complete, we can now categorize it as a **Hilbert space**. The mathematical environment where quantum systems exist. Referring to our earlier analogy, this is the chessboard. Now the settings are established, we can move to detailing the pieces on the board.

1.2 Matrices

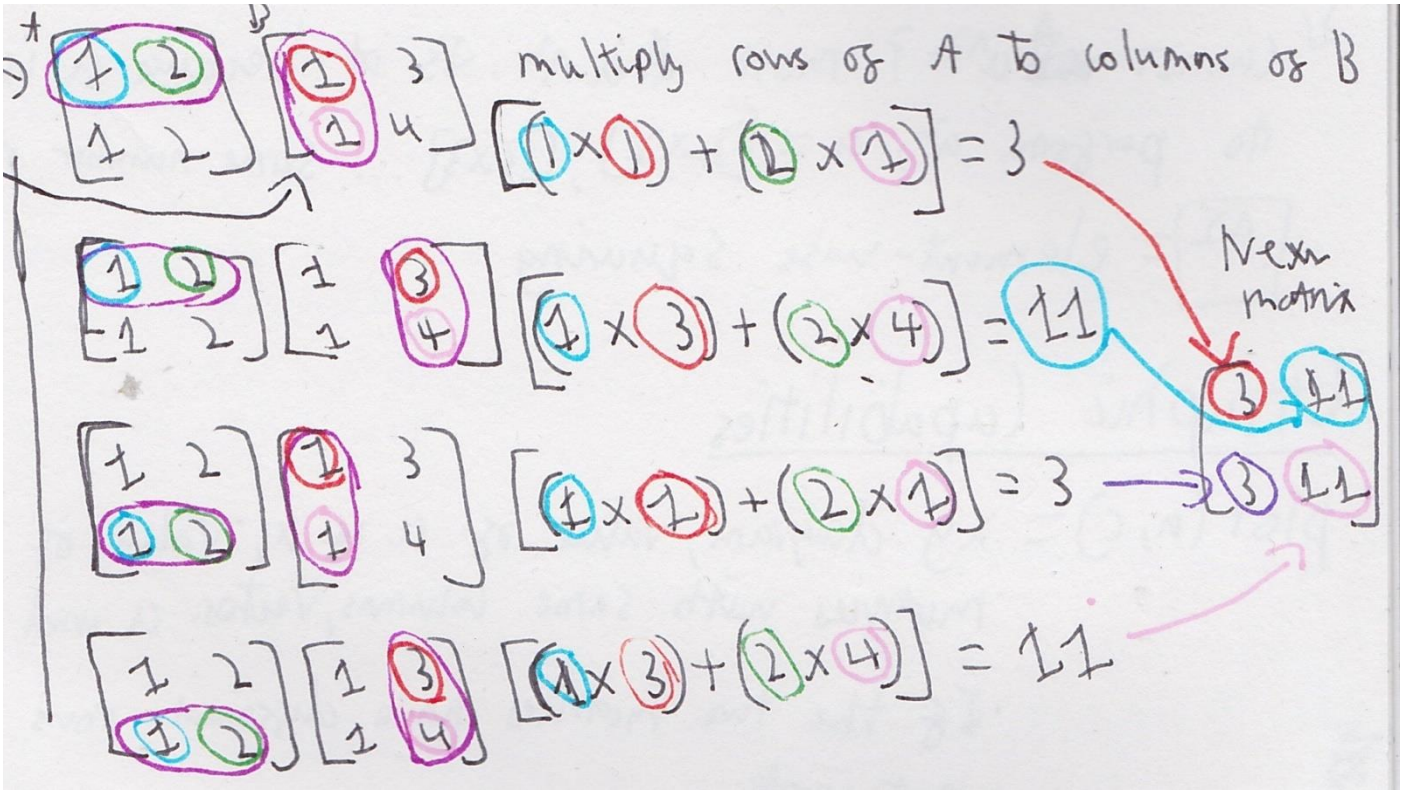
Matrices are rectangular arrays of elements - numbers, symbols, or expressions - arranged in rows and columns. They are essential for representing components like observables (2.6.3), quantum states (2.1), and Hermitian operators (2.5).

A matrix with m rows and n columns is called an m*n matrix. Row and column vectors are special cases, sized 1 * n and m * 1, respectively. We've already encountered matrices in the example of a vector space over $\mathbb{R}$.

1.2.1 Matrix Multiplication

Since we will see a lot of them interacting with one another throughout the report, it is important to outline how they do so.

$A = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$ $B = \begin{bmatrix} 1 & 3 \\ 1 & 4 \end{bmatrix}$ Below is the multiplication matrix A and B.



Not all matrices can be multiplied together, only ones where the column of one matrix must equal the row of the second matrix. For example, we cannot multiply A and C where $C = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \end{bmatrix}$.

1.2.2 Trace and Determinants for 2x2 Matrices

The trace of a matrix is the sum of its diagonal elements. In matrix A, with size n x n:

$\mathrm{Tr}(A) = \sum_{i=1}^{n} A_{ii}$ **For example** $\rightarrow A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then $\mathrm{Tr}(A) = a + d$

For the determinants, it is the multiplication of the top left diagonal elements - multiplication of the top right diagonal elements.

For example $\rightarrow A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then $\mathrm{Det}(A) = (a \times d) - (b \times c)$

You cannot calculate the determinant or trace of a single dimensioned vector , such as row or column vectors (1.1.2.2).

The determinant formula shown here is valid only for 2×2 matrices. Determinants of larger matrices require different techniques, such as Sarrus' Rule (for 3×3) or cofactor expansion.

2 Chapter 2: Quantum-specific Foundations

The body of this chapter will forge the concepts of quantum mechanics from the mathematical foundations found in the previous chapter. We will also introduce Dirac notation, a popular mathematical language used in quantum mechanics to represent quantum states and their inner products (as mentioned in 1.1.4)

2.1 Quantum States

A quantum system is a physical system (such as an electron or photon) whose state is described by a vector in a complex Hilbert space (1.1). These states (known as quantum states) encode all measurable information about the system and evolve according to the rules of quantum mechanics. They are the mathematical description of the system. Before we can build and interpret quantum states we must understand what a basis is and orthonormality.

2.1.1 Standard Bases

When measuring a quantum system, the outcome depends entirely on its current **quantum state**. To perform the measurement, we must select a **basis**—a set of reference states—onto

which the quantum state is projected. This choice defines the possible outcomes and their associated probabilities.

Choosing a basis is like selecting an alphabet: it determines how we express and interpret the system. Just as the same sound may appear differently in various alphabets, the same quantum state can be decomposed differently depending on the basis, influencing what we can "read" from it.

A basis for a vector space V must:

- Be Linearly Independent (no vector is a linear combination of others), and
- span V (every vector in V can be written as a combination of basis vectors).

In the case of a $\mathbb{C}^2$ space, a set $\{v1,v2\} \subseteq \mathbb{C}^2$ is a valid basis if:

- The equation $a_1v_1 + a_2v_2 = 0$, has only the trivial solution $a_1 = a_2 = 0$ (for $a_1,a_2 \in \mathbb{C}$ and $v_1,v_2 \in \mathbb{C}^2$), ensuring linear independence
- Any $v \in \mathbb{C}^2$ can be expressed as $v = (x * v_1) + (y * v_2)$ for $x,y \in \mathbb{C}$, confirming the set spans the space

We know these additions and multiplications listed above can be carried out without error due to $\mathbb{C}$, $\mathbb{C}^2$ adhering to the field axioms (1.1.1) and vector space axioms (1.1.2.1).

Let's test three potential bases for $\mathbb{C}^2$: $S_1 = \left\{\begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}2\\4\end{bmatrix}\right\}$ $S_2 = \left\{\begin{bmatrix}1\\0\end{bmatrix}\right\}$ $S_3 = \left\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right\}$

(row vectors since they are elements from a vector space (1.1.2.2))

Each quantum state in the basis is also referred to as 'basis states'.

linear independence holds when the only solution to $a_1v_1 + a_2v_2 = 0$ **is** $a_1 = a_2 = 0$

$S_1 : \left(-2 * \begin{bmatrix}1\\2\end{bmatrix}\right) + \left(1 * \begin{bmatrix}2\\4\end{bmatrix}\right) = 0$ - therefore S_1 fails linear independence
$S_2 : \left(0 * \begin{bmatrix}1\\0\end{bmatrix}\right) + (no\ second\ vector) = 0$ - therefore S_2 passes linear independence
$S_3 : \left(0 * \begin{bmatrix}1\\0\end{bmatrix}\right) + \left(0 * \begin{bmatrix}0\\1\end{bmatrix}\right) = 0$ - therefore S_3 passes linear independence

Spanning $v = (x * v_1) + (y * v_2)$, $x,y \in \mathbb{C}$ **and every vector** $v \in \mathbb{C}^2$

$S_1 : \left(x * \begin{bmatrix}1\\2\end{bmatrix}\right) + \left(y * \begin{bmatrix}2\\4\end{bmatrix}\right) = any\ vector\ in\ \mathbb{C}^2$ $\begin{bmatrix}2\\4\end{bmatrix} = 2\begin{bmatrix}1\\2\end{bmatrix}$ so this can be rewritten $(x + 2y)\begin{bmatrix}1\\2\end{bmatrix}$ – this equation will only produce vectors on the line $\begin{bmatrix}1\\2\end{bmatrix}$ and cannot reach vectors such as $\begin{bmatrix}2\\0\end{bmatrix}$ or $\begin{bmatrix}0\\2\end{bmatrix}$ for example. Therefore, S_1 fails spanning $\mathbb{C}^2$

$S_2 : \left(x * \begin{bmatrix}1\\0\end{bmatrix}\right) + \left(y * \begin{bmatrix}0\\0\end{bmatrix}\right) = any\ vector\ in\ \mathbb{C}^2$ - this can only produce vectors in the x-axis and the y-axis will always = 0. It cannot produce vectors such as $\begin{bmatrix}0\\2\end{bmatrix}$, $\begin{bmatrix}3\\2\end{bmatrix}$. Therefore, S_2 fails spanning $\mathbb{C}^2$

$S_3 : \left(x * \begin{bmatrix}1\\0\end{bmatrix}\right) + \left(y * \begin{bmatrix}0\\1\end{bmatrix}\right) = any\ vector\ in\ \mathbb{C}^2$ - this will produce the vector $\begin{bmatrix}x\\y\end{bmatrix}$, and since $x,y \in \mathbb{C}$, $\begin{bmatrix}x\\y\end{bmatrix}$ can represent any vector in $\mathbb{C}^2$. Therefore, S_3 succeeds in spanning $\mathbb{C}^2$

So now we can declare $S_3 = \left\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right\}$ is a valid basis for $\mathbb{C}^2$. There can be infinitely many valid bases for a given space. However, there is a term for a single, specific basis for every vector space. This is known as the 'Standard Basis'. They are named as such because they are used by default unless stated otherwise, so therefore are the usual standard. They also hold two additional properties.

The first is they are aligned with the coordinate axes, meaning each vector in the standard basis travels along only one axis in the coordinate system. So, as you may have noticed, as we have been discussing $\mathbb{C}^2$ – a two-dimensional vector space – S_3 holds this property with $\begin{bmatrix}1\\0\end{bmatrix}$ only travelling along the x-axis and $\begin{bmatrix}0\\1\end{bmatrix}$ travelling along the y-axis. The second property is orthonormality which is explained in section 2.1.3.

2.1.2 Dirac Notation

If $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$. (The basis from (2.1.1)).

- $|\rangle$ (pronounced ket) represents a quantum state vector, and the 0 and 1 inside is entirely arbitrary. A ket is a column vector, as we see above.
- $\langle|$ (pronounce bra) represent the conjugate transpose of the quantum state vector and becomes a row vector.

If $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$, then $\langle 0| = [1 \quad 0]$. This is equivalent to what we see happening in (1.1.4) with v and the conjugate transpose $\bar{v}$. When we see the bra-ket notation $\langle 0|1\rangle$ this represents

$[1 \quad 0] * \begin{bmatrix}0\\1\end{bmatrix}$

2.1.3 Orthonormality

This is the combination of orthogonality and Normalisation , and the condition stated mathematically is:

$$\langle u_i | v_j \rangle \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

Using the inner product equation (1.1.4.1), the top line (1 if I = j) refers to the Normalisation part, and the one below is the orthogonal part

2.1.3.1 Normalisation

This is the action of adjusting a vector so its length (or norm) = 1.

For vector $v \in \mathbb{C}^2$, its norm is $\|v\| = \sqrt{\langle v|v\rangle}$.

And then V normalised $= \frac{v}{\|v\|}$ (you divide the vector by its own norm)

This ensures that the probabilities of all possible outcomes sum to 1. In the context of quantum mechanics, these outcomes refer to the possible results of a measurement.

Example of Normalising a vector $v \in \mathbb{C}^2$

$v = \begin{bmatrix}3\\4\end{bmatrix}$ computing the norm: $\| v \| = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5$

V normalised $= \frac{v}{\|v\|} = \frac{1}{5} * \begin{bmatrix}3\\4\end{bmatrix} = \begin{bmatrix}0.6\\0.8\end{bmatrix}$

Now we can check if it sums to 1

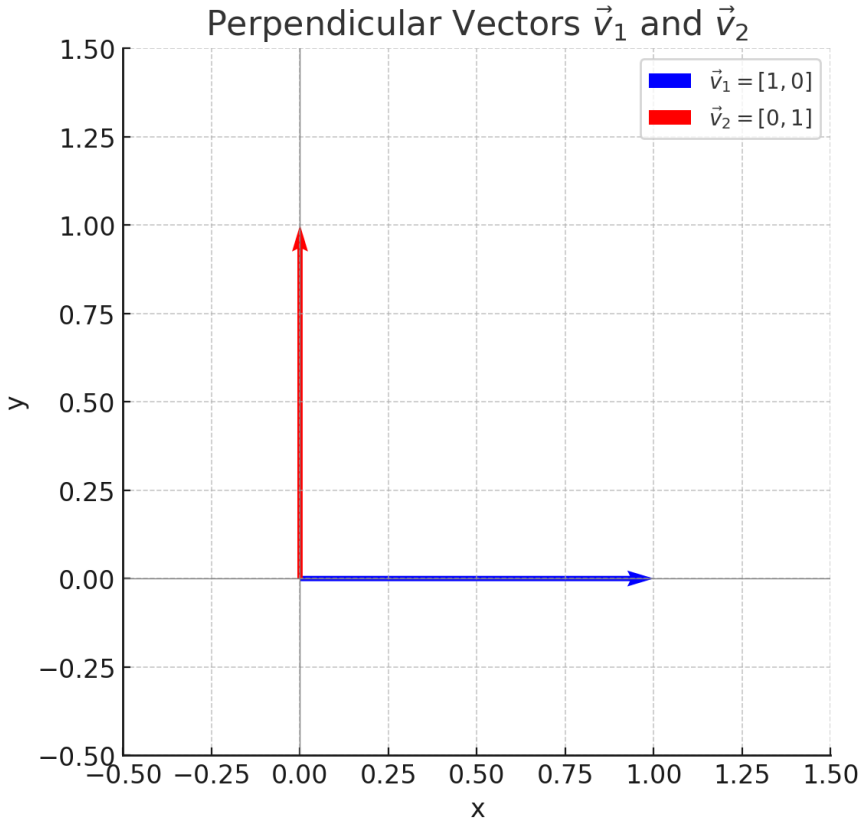
$< \text{Vnormalised} \ | \text{Vnormalised} >$

$[0.6 \quad 0.8] \begin{bmatrix}0.6\\0.8\end{bmatrix} = (0.6 * 0.6) + (0.8 * 0.8) = 0.36 + 0.64 = 1$

Normalisation is how we represent probabilities in quantum mechanics. This will become much clearer when we create and explore a full quantum system, since the system exists in different probabilities rather than a crisp answer.

2.1.3.2 Orthogonality

If $\langle v_i|v_j\rangle = 0$, $i \neq j$ and these two vectors are in the same vector space, they are perpendicular to each other. For example,



$$v_i = \begin{bmatrix}1\\0\end{bmatrix}, v_i = \begin{bmatrix}0\\1\end{bmatrix} \rightarrow \langle v_i|v_j\rangle = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = (1*0) + (0*1) = 0$$

Another way to think about this is if $\langle v_i|v_j\rangle = 0$, the vectors are effectively 'invisible' to one another. As mentioned in (2.1.1), a basis defines a set of reference directions for detecting or measuring a quantum state. But in this case, trying to detect v_j using v_i as a reference yields nothing, the inner product is zero which indicates no overlap. Physically, a measurement associated with v_i will never register a system prepared in state v_j.

This allows for other advantages, such as Clear state decomposition (Expressing a quantum state clearly in terms of basis states) and Simplified inner product calculations (1.1.4)

2.1.4 Superpositions

If the basis states are the ingredients, then a superposition is the recipe used to combine them and form a mathematical representation of a quantum system , which is a quantum state. Quantum states typically exist as superpositions of basis states — this is one of the defining features of quantum systems.

Unlike classical systems, which exist in a definite state at any given time, a quantum system can exist in multiple states simultaneously, described by a linear combination of basis vectors

from an n-dimensional Hilbert space (with n referring to the number of dimensions). This superposition is mathematically expressed as:

$$|\psi\rangle=\alpha_1|e_1\rangle+\alpha 2|e_2\rangle+\cdots+\alpha_n|e_n\rangle$$

- $|\psi\rangle$ = (ket psi) , a common symbol to represent a full physical state of a quantum system (another common symbol is $|\phi\rangle$ - pronounced 'phi')
- $\alpha_i \in \mathbb{C}$ (complex coefficients as probability amplitudes)
- $e_i \in \mathbb{C}^2$ (the orthonormal basis state)

Also remember, $|\psi\rangle \in \mathbb{C}^2$ too, since it is a requirement for the orthonormal basis to produce all vectors in the same space (2.1.1)

2.1.4.1 Complex Amplitudes and the Born Rule

Referring to (2.1.3.1), we must make sure that the Normalisation of the quantum state withholds. If we use $S_3 = \left\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right\}$ from (2.1.1) and create a quantum state with it

$$|\psi\rangle = \begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}$$

The basis vectors themselves are normalised (run process in 2.1.3.1), but the resulting quantum state $|\psi\rangle$ is **not** — its norm is:

$$\||\psi\rangle\|=\sqrt{1^2+1^2} = \sqrt{2} \neq 1$$

To make this a valid quantum state, we must normalize it, which leads us to introduce complex amplitudes. These are scalar coefficients (possibly complex) applied to the basis vectors so that the resulting quantum state satisfies the Normalisation condition.

In this case, we multiply each basis state by $\frac{1}{\sqrt{2}}$:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}$$

Now, the state is normalised:

$$\||\psi\rangle\|=\sqrt{\left|\frac{1}{\sqrt{2}}\right|^2 + \left|\frac{1}{\sqrt{2}}\right|^2} = \sqrt{\frac{1}{2}+\frac{1}{2}} = 1$$

But the significance of these amplitudes goes beyond Normalisation . In quantum mechanics, these coefficients - called complex amplitudes - carry physical meaning. The square of their

magnitudes gives the probability of observing the corresponding basis state when the system is measured. Therefore, the probability of measuring $|0\rangle$ (notation form 2.1.2) in:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \ , \qquad \text{Is } \frac{1}{2} = \left|\frac{1}{\sqrt{2}}\right|^2$$

This is known as the Born Rule. Where it states the probability of obtaining a specific measurement outcome is equal to the squared magnitude of the complex amplitude associated with that outcome. This rule connects the mathematics of quantum states to actual measurement outcomes.

Therefore, a quantum state describes the physical system in quantum mechanics, mathematically represented as a vector in a complex Hilbert space. To interpret or manipulate this state, we express it in terms of a chosen basis — most commonly the standard basis, which is orthonormal, and each with their respected complex amplitudes . In essence, understanding quantum states allows us to represent and manipulate quantum mechanics.

2.1.4.2 Pure and Mixed States

A pure state is a quantum system that is in a definite, fully known state represented by a single state vector $|\psi\rangle$ in a Hilbert space. Mathematically, a pure state in 2-dimensional Hilbert space is:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \text{ where } |\alpha|^2 + |\beta|^2 = 1$$

The density matrix is equal to its outer product (1.1.4.2): $p = |\psi\rangle\langle\psi|$. And satisfies (1.2.2): $\mathrm{Tr}(p^2) = 1$

The density matrix (or density operator) ρ is a mathematical object that represents the state of a quantum system is a crucial component when calculating measurement probabilities (3.6.2)

A mixed state is not a superposition, but a statistical mixture of pure states. Meaning instead of one definite quantum state (a pure state), it is a collection of possible pure states, each with classical probability.

It describes a system which we only have partial knowledge of.

An example is a quantum system that is randomly in either $|0\rangle$ or $|1\rangle$ with a 50% chance each. We are unsure which one at any stage, which means:

- no quantum inference. Since the system is not in a coherent superposition, like $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, we cannot observe interference effects that depend on the relative phase between components. Leading to the system behaving more like a classical coin toss than a quantum state.

- phase relationship between the two components. In a superposition, the relative phase (e.g., the difference between $|0\rangle + |1\rangle$ and $|0\rangle - |1\rangle$)affects measurable outcomes. In a mixed state, there is no well-defined phase between the states in the mixture, so such effects simply don't occur.

Mathematically, it is defined as:

$$p = 0.5|0\rangle\langle 0| + 0.5|1\rangle\langle 1| = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

Therefore, $\mathrm{Tr}(p^2) < 1$

Our entire buildup of the foundations of quantum mechanics has been entirely abstract and void of physical results we can see in our own 3 dimensions. So now I believe would be a good time to represent and discuss a well-known experiment that is extremely simple to recreate and acts out all the parts we have been discussing. Once we see this physical experiment, we can move onto how we represent quantum systems computationally.

2.2 Qubits

A qubit is short for quantum bit, and as the name suggests is the equivalent of a bit in quantum computation. A qubit is both a quantum system and the fundamental unit of information in a quantum system. Key differences between qubits and classical Bits are:

Properties	**Classical Bit**	**Qubit**
States:	0 or 1	Exists in a superposition
Representation:	Binary, Hexadecimal, Boolean	Complex Vector Space (Hilbert spaces)
Measurement:	Deterministic	Probabilistic (collapses to deterministic when measured)
Operations:	Logic Gates(AND, OR, NOT)	Quantum Gates (Such as Pauli-X, Pauli-Y, Pauli-Z)

Qubits are what we observe changing state and evolve over time or when transformed by an operator. In this simulation, the qubit is the ‘test subject’ as you will.

2.3 The Three Polarizer Paradox

Quantum A perfect example to show quantum states in our 3-dimensional reality is the 'Three Polarization experiment'

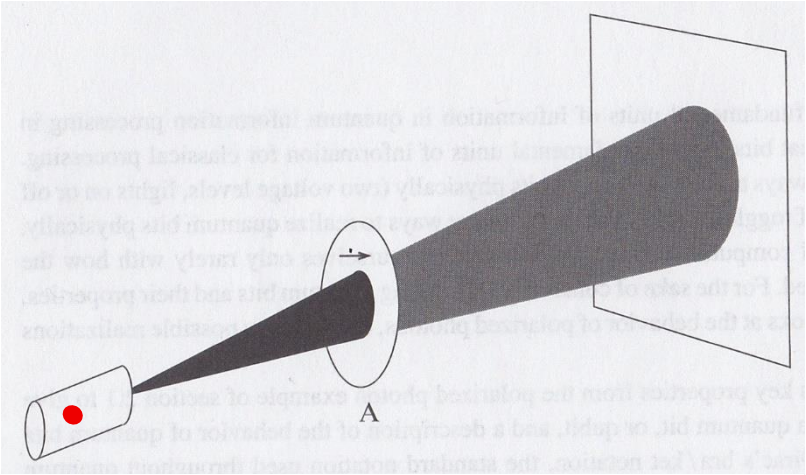


Figure 1.1

To Begin with, we see a beam of light being produced from a light source (indicated with a red circle) passing through polaroid A and then hitting and ending at a wall. Polaroid A polarizes the photons in the light beam to quantum state $|\rightarrow\rangle$.

(Rieffel and Wolfgang Polak, 2011) "Polaroid A attenuates unpolarized light by 50%"

$|\rightarrow\rangle$ represents a quantum state vector. Exactly like $|\psi\rangle$. The symbol in the middle is completely arbitrary.

Now, if we add a 2nd polaroid, named Polaroid C, with the polarization of $|\uparrow\rangle$ (perpendicular to A), all light is blocked as seen in figure 1.2. All photons are blocked and none reach the wall. Now, adding a third polaroid to figure 1.2 should make no difference, however we see a major effect when we do.

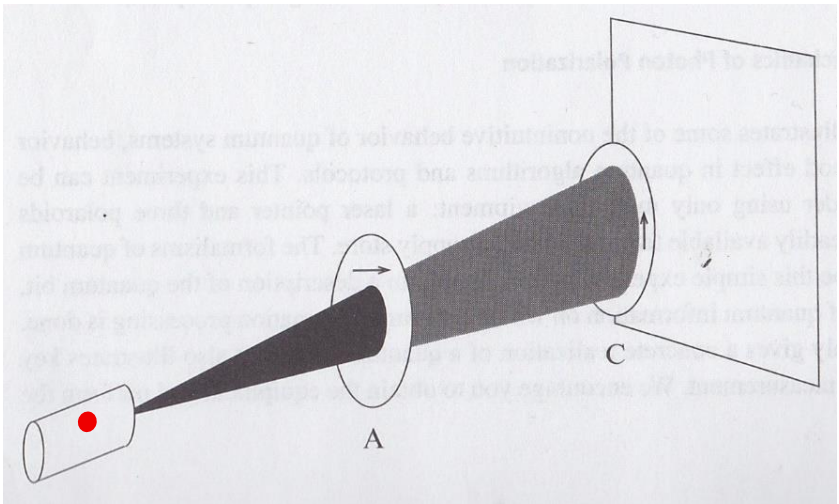


Figure 2.2

If we add polaroid B in between A and C, with the polarization of $|\nearrow\rangle$ which is an equal 45-degree angle between A and C (figure 1.4) Some number of photons now reaches the wall at the back. How is this possible? Let us explain this in terms of vectors.

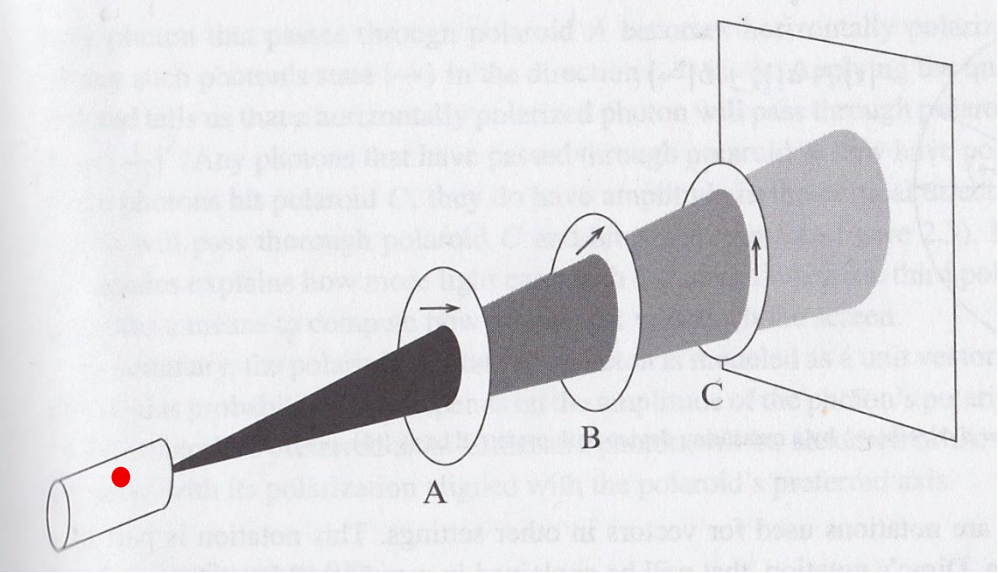
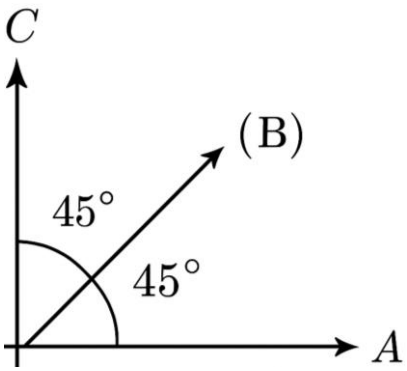


Figure 3.3

A = $|\rightarrow\rangle$ (horizontal polarization)

C = $|\uparrow\rangle$ (vertical polarization)

B = a$|\rightarrow\rangle$ + b$|\uparrow\rangle$. (diagonal polarization)



Photons passing through A are polarized to $|\rightarrow\rangle$, which provides them a complex amplitude of zero to the vertical polarization ($|\uparrow\rangle$). Photons passing through A have the quantum state $|\nearrow\rangle = 1|\rightarrow\rangle + 0|\uparrow\rangle$, which can be simplified to $|\rightarrow\rangle$. If we define $|\rightarrow\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|\uparrow\rangle = \begin{bmatrix}0\\1\end{bmatrix}$, then:

$$\langle\uparrow|\rightarrow\rangle = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = (0*1)+(1*0) = 0$$

Vectors $|\rightarrow\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|\uparrow\rangle = \begin{bmatrix}0\\1\end{bmatrix}$ lie along orthogonal axes, representing horizontal and vertical polarization. Inserting Polaroid B at 45° projects photons into a superposition:

$$|\nearrow\rangle = \frac{1}{\sqrt{2}}|\rightarrow\rangle + \frac{1}{\sqrt{2}}|\uparrow\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$$

This equal-amplitude state is normalised and reflects the geometric midpoint between the two axes. As a result, the probability of measuring $|\uparrow\rangle$ becomes $\left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$, allowing half the photons to pass through Polaroid C—explaining why some light reaches the wall.

A to B:

- A polarizes the photons into state $|\rightarrow\rangle$
- Probability of passing through B (projecting (2.6.3.1) onto $|\nearrow\rangle$) – using the inner product equation.

$$p_{A\rightarrow B} = |\langle \nearrow | \rightarrow \rangle|^2 = \frac{1}{2}$$

B to C:

- Photons are now in state $|\nearrow\rangle$
- Probability of passing through C

$$p_{B\rightarrow C} = |\langle \uparrow | \rightarrow \nearrow \rangle|^2 = \frac{1}{2}$$

Remembering that only 50% passes through A initially total probability of a photon reaching the wall is $\frac{1}{2}$ x $\frac{1}{2}$ x $\frac{1}{2} = \frac{1}{8} = 12.5\%$. We should expect to see 12.5% of the photons that pass-through from the light source to reach the wall. Which is consistent with the analysis in Chemistry LibreTexts (2019) "12.5% of the light illuminating the first vertical polarizer passes the final horizontal polarizer."

This experiment illustrates how a quantum system evolves through measurement, with Polaroid B inducing a superposition of basis states characterized by complex amplitudes. As Rieffel and Polak (2014) note, any normalised state like $a|\rightarrow\rangle + b|\uparrow\rangle$ is a valid qubit. This encapsulates key quantum principles—superposition, amplitude, and measurement—and shows how physical phenomena map directly to computational models of qubits.

2.3.1 Computational Basis States

The computational basis consists of the states:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

These form the standard basis for qubits and allow the construction of arbitrary quantum states through superposition. Unless stated otherwise, these are the default representations for $|0\rangle$ and $|1\rangle$.

2.3.2 Hadamard Basis States

Beyond the computational basis, qubits can also be expressed in alternative bases, as Hilbert spaces admit infinitely many valid bases (2.1.1). widely used example in quantum computing is the **Hadamard basis**, defined by:

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

These states resemble the superpositions introduced in the Three Polarizer Experiment (2.2) and represent equal mixtures of the standard basis states. They also form a valid orthonormal basis for the same complex vector space.

2.3.3 Y-Basis States

The third set of what is commonly known as the "three principal (or standard) bases" in quantum computing are known as the Y-Basis States, and are as follows:

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

(where i is the imaginary unit) . These states also form a valid orthonormal basis for the same Hilbert space. While the computational basis is often labelled the **Z-basis**, and the Hadamard basis the **X-basis**. The significance of these labels will become clearer in the context of Bloch sphere visualizations (see Section 2.5.2).

2.3.4 Superconducting Qubits (Two-level System)

These three bases allow us to construct superpositions that fully describe the state of a qubit. A qubit exists in a two-dimensional complex vector space, meaning its quantum state can be expressed as a linear combination of two basis states. The choice of basis is flexible, if the states are orthonormal and span the space.

This formalism is particularly appropriate for superconducting qubits, where the physical system may have many energy levels, but only the lowest two are used for computation. These two energy levels form the effective qubit, making it a genuine two-level quantum system for practical purposes.

We will now refer the three bases:

- $|0\rangle$ and $|1\rangle$ as Z-Basis
- $|+\rangle$ and $|-\rangle$ as X-basis
- $|\mathrm{i}\rangle$ and $|-\mathrm{i}\rangle$ as Y-basis

2.4 Rabi Oscillations

Rabi oscillations describe the coherent population oscillations of a two-level quantum system when it is driven by an external oscillating field, such as a laser or microwave pulse. In our simulation, this is the physical phenomenon underlying qubit evolution.

The oscillation rate is characterized by the Rabi frequency, which quantifies how rapidly the quantum state transitions between its two levels. This frequency is measured in radians per second, and in quantum computing contexts, it is often expressed in units of π to simplify time evolution cycles.

2.5 Hamiltonians

A Hamiltonian represents the total energy of a system and governs how the quantum state (like qubits) evolves over time. Mathematically, there are known as 'Hermitian operators' and are represented using matrices **(1.2 Matrices)**. These types of operators represent observables – measurable physical quantities of a quantum system (such as a qubit). We will be using Hermitian operators (a mathematical object that acts on a quantum state to produce another state) to conduct time evolution onto our qubits; the ones we will utilize are called Pauli Gates.

2.5.1 Pauli Gates

A specific set of 2x2 matrices **(1.2 Matrices)** that are used for matrix manipulation and simulating qubits changing state. They are equivalent to logic gates on classical bits

$$\text{Pauli} - \text{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \sigma_x \qquad \text{Pauli} - \text{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \sigma_y$$

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

The Pauli – X gate is the equivalent to a classic NOT gate. They flip $|0>$ to $|1>$ and vice versa. Example: $\sigma^X * |0> = |1>$

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{pmatrix} (0 \cdot 1) & (1 \cdot 0) \\ (1 \cdot 1) & (0 \cdot 0) \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$ = Pauli-X gate transforming the qubit.

These matrices will be used to build the driving Hamiltonians, which govern time evolution of qubits. The aim is to use all three.

2.5.1.1 Type of effect on standard Bases and Phases

Each Pauli matrix may also introduce a global or relative phase when transforming the qubit bases, which hold different physical implications. Where one or more can occur:

- Bit flipping – the qubit switches between basis states (as seen in the example in (2.5.1))
- A global phase (like multiplying a state by i) does not affect the measurement outcomes.

- A relative phase between components in a superposition does affect interference and is physically meaningful.

For example, if we use the Hadamard basis $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and a Pauli matrix transforms the state into $i\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, this is just the original state multiplied by the global phase i: $i \times |+\rangle$.

- Measurement outcomes do not change.
- Physically indistinguishable from the original state

Whereas, if $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is transformed to $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, only $|1\rangle$ has been multiplied by i (the complex amplitude of the state has been affected). Meaning the phase between $|0\rangle$ and $|1\rangle$ have changed:

- Does affect measurement outcomes
- New state, it becomes equal to $|i\rangle$ from the y-basis state

Bit flipping can occur on its own or alongside a global or relative phase shift, depending on the transformation applied to the qubit.

2.5.2 Driving Hamiltonians

These are the functions that will be put into effect to evolve the qubit states over time.

$$H = \Omega * \sigma$$

The choice of Pauli matrix determines which basis states the qubit will oscillate between. Each Pauli operator corresponds to a rotation axis on the Bloch sphere (a 3D visual graph to plot qubits (section 2.7 – Bloch Spheres).

- As seen in the example in (2.5.1) the σ_x drives transitions between the computation basis states $|0\rangle$ and $|1\rangle$
- The rate of oscillation is determined by the Rabi frequency Ω.

These principles will be tested and visualized during simulation in Chapter 4, where the effect of each Pauli matrix on the standard basis states will be explored.

2.6 Time Evolution

Time evolution describes how a quantum system changes with time according to its governing physical laws. In quantum mechanics, this evolution is dictated by the system's Hamiltonian, which acts as the generator of time-dependent dynamics. Depending on whether a system is closed (isolated) or open (interacting with an environment), different mathematical frameworks are used to model its behaviour. This section introduces the key tools used to describe time evolution, including unitary evolution for closed systems, collapse operators for open systems, and observables such as projectors and Pauli operators, which allow us to interpret measurements and probabilities as a system evolves.

2.6.1 Closed Systems

If we only pass through the Hamiltonian, Initial and time limit array variables , this will be efficient to simulate a closed system (no unknown external forces influencing the quantum system we observe). The equation the function uses in this case is the state vector evolution.

Schrödinger Picture

$$|\psi(t)\rangle = e^{-iHt}|\psi_{inital}\rangle$$

$|\psi_{initial}\rangle =$ the state the system is assumed to be in at time 0

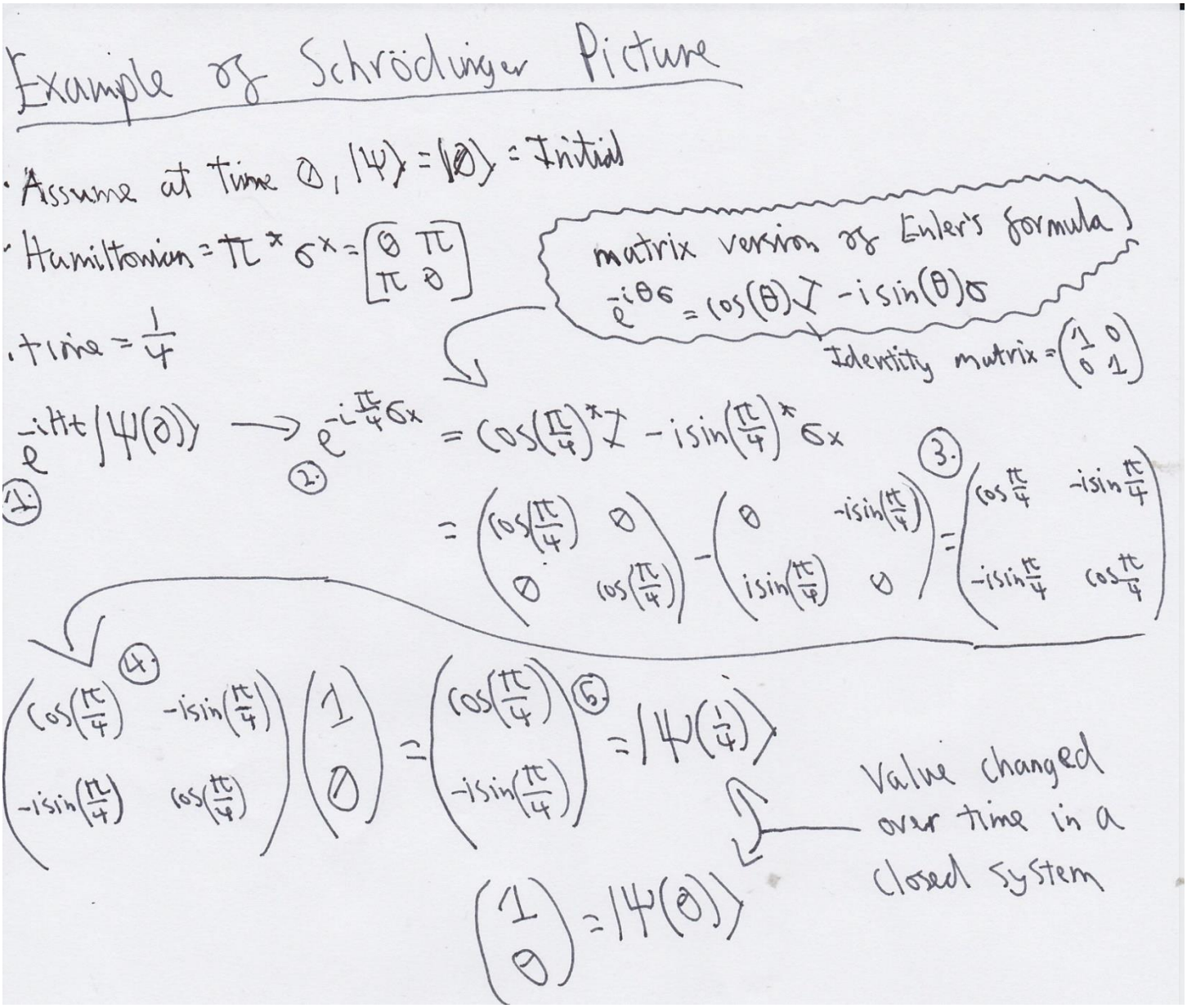
$|\psi(t)\rangle =$ quantum state at time t

H = Hamiltonian

e = Euler's Number

i = imaginary number ($\sqrt{-1}$)

Here is an example of how the equation shows a quantum system evolving over time.



$\begin{pmatrix} \cos\left(\frac{\pi}{4}\right) \\ -i\sin\left(\frac{\pi}{4}\right) \end{pmatrix}$ can be simplified to $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix}$

We have now shown a mathematical example of a quantum state evolving over time in a closed system. This is a crucial calculation and allows us to simulate closed systems.

2.6.2 Open Systems (Collapse Operators)

This variable represents decoherence of the quantum system, which is rate in which superconducting qubits (the ones that are being simulated) lose quantum coherence. When referring to coherence for a qubit, this is the ability for the system to remain in a superposition, such as $|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$. Decoherence is the system interacting with its environment, and loses

quantum information and thus, the ability to exist in a superposition of states. the final state of the system then becomes a mixture of classical deterministic values. For example,

$$P = |\psi\rangle\langle\psi| = \begin{pmatrix} |\alpha|^2 & \alpha\bar{\beta} \\ \bar{\alpha}\beta & |\beta|^2 \end{pmatrix} \rightarrow P(t) = \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix}$$

P shows the outer product (reference) of $|\psi\rangle$, and $P(t)$ is this quantum system at a certain time t after interacting with its environment and losing coherence ($\alpha\bar{\beta}$ and $\bar{\alpha}\beta$). These two values represent the phase relations between $|0\rangle$ and $|1\rangle$, which allows this system to behave quantum mechanically.

Decoherence is the greatest threat in quantum computing, and great amounts of effort is spent to decrease the environmental effect as much as possible. if this parameter is empty, this represents a closed (or pure) system with no decoherence.

2.6.3 Observables

(refer to 2.4 Hamiltonian) Observables are operators to extract physical information from the qubit that has evolved over time. The equation that is then calculated is the Expectation Value Equation – EVE (also the general form of the Born rule in density matrix formalism).

$$\langle O(t)\rangle = \mathrm{Tr}[\rho(t) \cdot O]$$

Tr[] = Trace Operation (1.2.2 Matrix Trace)

O = observable – the operator to represent a measurable physical property. As mentioned in (1.2.1 Standard Bases), this essentially defines the measurement that is performed by projecting the quantum state onto the observable.

$p(t)$ = The density matrix (2.1.4.2) of the evolved quantum system (qubit) at time t.

$p(t)$ is the outer product of the evolved state from the Schrödinger picture (Section 2.6.1):

$$p(t) = |\psi(t)\rangle\langle(t)\psi|$$

The Schrödinger picture computes the quantum state's evolution, while the **expectation value equation (EVE)** determines the probability of finding the system in a given state.

The result $\langle O(t)\rangle$, or expectation value, depends on the chosen observable O. To illustrate the two main observable types, we use the states $|\psi(0)\rangle$ and $|\psi(\frac{1}{4})\rangle$ from the closed-system example in Section 2.5.1.

2.6.3.1 Projectors

As mentioned in (1.2.1 Standard Bases), a basis must be used to measure the quantum state by projecting it onto said basis. The chosen basis is known as a projector, and in this case the observable can be a projector. For example, A projector onto $|0\rangle$ is created by:

$$\text{Observable} = P_0 = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

the excited state refers to the state we are measuring the probability for, while the ground state is the system's initial state before measurement or interaction This operator projects any state onto $|0\rangle$ and gives the probability of finding the system in state $|0\rangle$ when measured, setting $|0\rangle$ as the excited state and naturally implying $|1\rangle$ is the ground state.

Let's run the calculations using $O = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$

For $|\psi(0)\rangle$:

$$p(t) = |\psi(0)\rangle\langle(0)\psi| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \longrightarrow \text{Tr}\left[\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\right] = \text{Tr}\left[\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\right] = 1$$

For $|\psi(\frac{1}{4})\rangle$:

$$p(t) = |\psi(\tfrac{1}{4})\rangle\langle(\tfrac{1}{4})\psi| = \frac{1}{2}\begin{pmatrix} 1 & i \\ -i & 1 \end{pmatrix} \longrightarrow \text{Tr}\left[\frac{1}{2}\begin{pmatrix} 1 & i \\ -i & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\right] = \text{Tr}\left[\frac{1}{2}\begin{pmatrix} 1 & 0 \\ -i & 0 \end{pmatrix}\right] = \frac{1}{2}$$

We see this qubit with the initial state of $|0\rangle$ evolving over time by a Hermitian operator $\pi\sigma_x$ to perform rabi oscillations in a closed system. At time 0, the qubits state = $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ (which is $|0\rangle$) and the probability of finding this qubit in state $|0\rangle$ is 1. At time $\frac{1}{4}$, the qubit state = $\begin{pmatrix} \cos(\frac{\pi}{4}) \\ -i\sin(\frac{\pi}{4}) \end{pmatrix}$ and the probability of finding this qubit in state $|0\rangle$ is $\frac{1}{2}$. This is the data that will be used to create 2D graphs to visualize results.

2.6.3.2 Pauli Operators

We can also set the Observable to be a Pauli operator. This produces an expectation value of the spin-i component at time t (i = x,y,z depending on which Pauli matrix is used).

Using the Pauli operators σ_x, σ_y and σ_z we can measure the qubit states along an X,Y and Z axes, providing us the appropriate number of dimensions to visualize a geometric graph (see section 2.7)

The equation would be simply (using Pauli-x operator as an example):

$$\langle\sigma\rangle = \mathrm{Tr}[\rho(t) \cdot \sigma_i] \quad \text{for } i = (x, y, z)$$

p(t) acts as the exact same, but instead of an outer product of a specific state we want a probability for we simply use a Pauli matrix.

In this case, the result is not a probability, but a measure of a physical trait of the qubit.

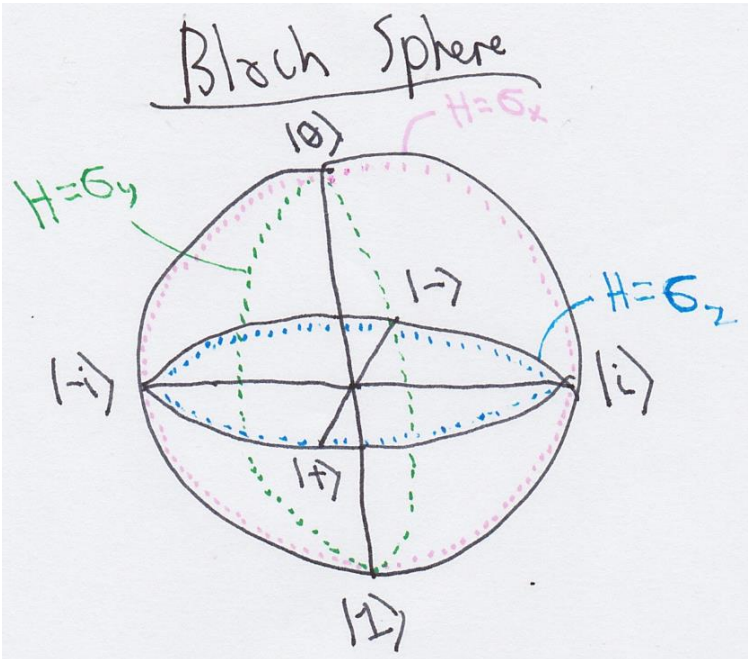
The possible values range between [+1,-1] for example on the z axis (using σ_z).

- $\langle\sigma\rangle$ = +1 implies fully aligned with state $|0\rangle$
- $\langle\sigma\rangle$ = -1 implies fully aligned with state $|1\rangle$
- $\langle\sigma\rangle$ = 0 implies equal superposition between outcomes ($\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$)

We can simulate the qubit physically moving around the z axis, representing the oscillations between states. Using this method, we will produce results for a 3-dimensional showcase.

2.7 Bloch Spheres

The **3D Bloch sphere** is a geometric tool for visualizing the full quantum state of a single qubit in a complex Hilbert space. While 2D graphs (Section 3.6.1) display measurement probabilities over time, the Bloch sphere offers a more intuitive, spatial representation of state evolution.



Building on Hilbert space and qubit concepts (Sections 2.1.6 and 2.3), the sphere maps any pure qubit state to a point on the surface of a unit sphere. The poles represent the computational basis states |0⟩ and |1⟩; all other points correspond to their superpositions. As detailed in Section 2.1.4, the relative phase and magnitude of complex amplitudes determine each state's position.

This model is especially useful for visualizing **Rabi oscillations** (Section 2.4), where the qubit undergoes coherent rotations around an axis defined by the Hamiltonian (Section 2.5). The rotation direction aligns with the axis of the driving Pauli operator: σ_x, σ_γ, or σz.

The Bloch sphere applies only to **single-qubit systems**. Pure states lie on the surface, while **mixed states** (Section 2.1.4.2) appear inside, shifting toward the centre as coherence is lost (Section 2.6.2).

In essence, the Bloch sphere bridges abstract linear algebra and spatial intuition, making it a vital tool for understanding single-qubit dynamics.

3 Chapter 3: Component Description

This chapter will list and explain the tools and environments used for simulation and visualization. It serves as a bridge between the mathematical and quantum mechanical foundations established in the previous two chapters and their translation into computational design. This prepares the groundwork for Chapter 4, where the development and implementation of the simulation program are carried out.

3.1 Computational Environment

Python was chosen as the primary programming language for this project due to its widespread adoption in scientific computing, its clean and readable syntax, and the extensive availability of specialized libraries. IDLE was selected for the IDE, due to its simplicity and inbuilt standard Python Installations. This allows for focused development without the overhead of more complex environments, while remaining sufficient to support script-based computation with the required Python libraries

3.1.1 Core Libraries

The script uses three main libraries, each supporting a key aspect of the simulation: numerical computation, quantum system modelling, and data visualization. Their built-in functions and predefined tools streamline implementation, reduce development time, and ensure accurate simulation of complex quantum behaviour.

To maintain clarity and efficiency, only essential components are imported. Unused methods are excluded to reduce overhead and keep the code focused on the simulation's specific needs.

3.1.1.1 Numpy

A numerical computing library providing support for arrays and linear algebra. Providing easier and efficient matrix transformations as well as producing an array of time stamps.

3.1.1.2 Matplotlib

A numerical computing library that supports array manipulation and linear algebra operations. It enables efficient matrix transformations and facilitates the creation of numerical sequences, such as arrays of time stamps, which are essential for simulating time evolution in quantum systems.

3.1.1.3 QuTiP

QuTiP (Quantum Toolbox in Python) is a library for simulating quantum system dynamics. It uses quantum objects (Qobj) to model components like states and operators. QuTiP supports time evolution, state transformations, and expectation value calculations, and handles both closed and open quantum systems. It also provides visualization tools—including Bloch spheres and probability graphs—that integrate smoothly with libraries like Matplotlib.

3.2 Representation of Quantum States in code

This simulation will be representing superconducting qubits in a Hilbert space ($\mathbb{C}^2$); therefore, we will represent qubits using the three principal bases in Dirac notation (2.1.2):

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

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

These also define the qubits initial values when time = 0 (the beginning of the simulation)

3.2.1 Qubits as Quantum Objects

The function from QuTiP (3.1.1.3) named ‘basis’ allows us to create a quantum state. The syntax is :

Basis(N, T)

N = the dimensions of the Hilbert space. This also defines the number of basis states there are.

T = index of the basis state you want created. 0 <= T < N.

Therefore, to run an example:

Basis(2,0) = $\begin{bmatrix}1\\0\end{bmatrix}$ = $|0\rangle$ or Basis(2,1) = $\begin{bmatrix}0\\1\end{bmatrix}$ = $|1\rangle$.

These are all the states that can be produced using Basis(2,i). It also produces the vector space in ket type (meaning a column vector). To represent the vector in Bra type (a row vector) we can write:

Basis(N, T).dag() = converts the vector into row form . Therefore Basis(2,0).dag() = [1 0]

This is essential for representing :

- Inner products (1.1.4) = Basis(N, T).dag() * Basis(N, V) (this returns a scalar value)

- Outer product (1.1.4.2) = Basis(N, V) * Basis(N, T).dag() (this returns a matrix)

As stated in (1.1.4.2), if the qubits are pure states (as they will be using the three principal bases) then outer product = density matrix.

The outer product result (a matrix) is also a Qobj (3.1.1.3).

3.2.2 Complex Numbers

Python has an inbuilt function to represent imaginary unit i, which is "1j". A complex number can be then initialized by :

Complex_number = 5 + 2j

Or when creating a quantum state with a complex number, we can simply state:

Basis(N,M) + Xj where N,M,X $\in \mathbb{R}$ and j = $\sqrt{-1}$

3.3 Operators and Hamiltonians

Operators (Section 2.5) are described as Qobj instances in QuTip, just like quantum states. This reflects that quantum states (like $|0\rangle$) and operators(like Pauli-X,Y,Z) exist in the same Hilbert space. Meaning they can support all vector space operations and matrix manipulations, enabling simulations of state evolution, operator application, and observable measurements.

3.3.1 Pauli Operators

The Pauli matrices introduced in Chapter 2 are implemented in the simulation using QuTiP's built-in functions:

- sigmax() for the Pauli-X operator (σ_x)
- sigmay() for the Pauli-Y operator (σ_y)
- sigmaz() for the Pauli-Z operator (σ_z)

Each return a Qobj instance suitable for its use in creating a Hermitian operator, and can be applied directly in Hamiltonians

3.3.2 Driving Hamiltonians

As mentioned in (Section 2.4), the rabi frequency is expressed in units of pi, therefore the constant np.pi will be used.

Ω = C * np.pi , where C $\in \mathbb{R}$ and Ω is the Rabi frequency .

Now we can use these components to produce a Hermitian operator.

$$H = \Omega * \sigma \text{ where } \sigma \in \{\text{sigmax}(), \text{sigmax}(), \text{sigmax}()\}$$

This Hamiltonian governs the time evolution of the qubit we construct, and determines the rotation of the qubit on a Bloch sphere (see section 2.7 and 3.6.2)

3.4 Time Evolution Simulation

QuTip provides an essential function named 'mesolve' that can calculate the Schrödinger equation in section 2.6.1 and reproduce rabi oscillations (section 2.4). Due to the lack of the collapse operator input, this simulates a closed system (see section 2.6.1)

result = mesolve(H, Initial, Tlist, [], [])

- H = the Hermitian operator (section 3.3.2)
- Initial = quantum state we want to see evolve (section 3.2)
- Tlist = an array of time points at which the simulation will evaluate and return the system's state.

Numpy (3.1.1.1) provides a useful tool for producing an array. The syntax as follows:

Array = np.linspace(U, V, M) , where:

U = the starting point of the interval

V = ending point of the interval

M = number of equally spaced points to generate between U and V

For example, np.linspace(0, 2, 400) produces 400 evenly spaced points between 0 and 2.

The time units used here are arbitrary and, as will be demonstrated in Chapter 4, measuring qubit evolution in terms of time is not always the most effective approach

3.5 Observables and Measurement

Referring to Section 2.6.3 on observables, these operators are also essential computationally, as they are used to extract measurable quantities from the simulation. Specifically, observables are required to generate data that can be visualized in plots, allowing us to observe how the quantum state evolves over time.

In the mesolve equation mentioned earlier, the observable (denoted as O), is inputted by:

Results = mesolve(H, Initial, Tlist, [], O) .

O be a single operator or a list of operators, allowing multiple observables to be tracked at once. When a list is provided, mesolve computes and stores the expectation values for each observable over time, resulting in multiple datasets simultaneously available through results.expect[i].

3.5.1 Projectors

As shown in Section 2.6.3.1, operators are produced via the outer product of the same state, there we can use the functions outlined in 3.2.1:

Observable = basis(N, V) * basis(N, V).dag()

This is a projector onto basis state |V⟩ in an N-dimensional Hilbert space.

It must be in this formation to represent $|0\rangle\langle 0|$.

(this will only work for pure states, refer to section 2.1.4.2)

3.5.2 Pauli Expectation Values

To produce expectation values (detailed in 2.6.3) we simply set

$O \in \{\text{sigmax}(), \text{sigmax}(), \text{sigmax}()\}$

3.6 Visualization of Results

I plan to produce 2 main types of results, a 2D and a 3D graph to represent the outcomes from the program. Both use the matplotlib library (2.1.2 Matplotlib), however the 3D graph class is provided by the qutip library (2.1.3 Qutip), but it is processed and rendered by the matplotlib library. These graphs will be the only values returned to the user.

3.6.1 2D XY Graphs

A 2D line graph effectively visualizes how a quantum system's measurement probability evolves over time. The X-axis represents simulation time, and the Y-axis displays the corresponding probability at each time point.

As explained in Section 2.6.3.1 (Projectors), these probabilities are calculated using projectors to determine the likelihood of the system being in a specific state at time t. Plotting these values over time reveals how measurement outcomes vary as the system evolves.

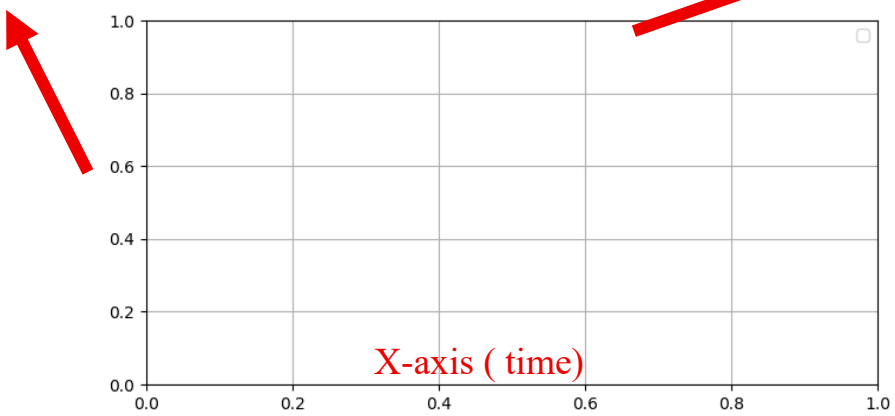
Each graph shows the probability of measuring the system in a particular target state. Multiple graphs can compare different outcomes simultaneously during the evolution.

These plots are generated using Matplotlib, with the following syntax:

1. plt.figure(figsize=(x, y)) where x,y ∈ ℝ
2. plt.plot(Tlist, Results, label="Data label")
3. plt.xlabel("X-axis Title")
4. plt.ylabel("Y-axis Title")
5. plt.title("Graph Title ")
6. plt.grid(True)
7. plt.legend()
8. plt.show()

1. determines the size of the graph (in inches) when appearing in the new window.
2. Plots the data using Tlist as the X-axis and Results as the Y-axis
3. Name of the X-axis
4. Name of the Y-Axis
5. Adds a Title for the whole Graph
6. Provides a background grid (not required but good practice for scientific grids)
7. displays the legend using the 'Date label' provided earlier (not required but good practice for multiple datasets)
8. Renders and displays the Graph

Y-axis (measurement probabilities)



The data is plotted here to show the evolution of the value of the measurement probability over time

3.6.2 3D Bloch Sphere Graphs

The orientation of the Bloch vector is directly tied to **expectation values of the Pauli operators** (Section 3.5.2). Specifically, the coordinates (x,y,z) of the Bloch vector at any time t are given by the expectation values of (by using the equation in Section 2.6.3):

$$x = \langle \sigma x(t) \rangle \qquad y = \langle \sigma y(t) \rangle \qquad z = \langle \sigma z(t) \rangle$$

These values are calculated using QuTiP's mesolve() function, where the Pauli matrices act as observables (see Section 2.6.3.2). This mapping provides a complete description of a qubit's states at any time.

1. Bloch = Bloch()
2. Bloch.point_color = [C]
3. Bloch.add_points([X-values, Y-values, z-values]
4. Bloch.title = [Graph Title]
5. Bloch.show()

1. initializes Bloch sphere object
2. Sets the point colour(s); C can be a string or list.
3. Adds points using X, Y, Z coordinates (representing qubit states on the unit sphere).
4. Sets the plot title.
5. Renders and displays the Bloch sphere.

Though matplotlib is not directly needed just to show the Bloch sphere, but it is required behind the scenes, since QuTiP uses Matplotlib to generate the 3D figure.

Now we have listed all components required to produce a sufficient simulation with encouraging result

4 Chapter 4: Program Development

The chess match can now begin. The board has been set up, each chess piece examined, and the legal moves for each clearly understood. With this groundwork in place, we can now begin constructing a strategy aimed at producing a desired outcome — or, at the very least, an outcome that can be meaningfully analyzed.

4.1 Project Aims and Objectives

Our goal is to simulate rabi oscillations in a Hamiltonian-driven two-level quantum system. The quantum system we will be using will be a qubit, specifically a superconducting qubit. Once the calculations are complete, we will model the results using appropriate graphical representations (see Section 3.6).

This phenomenon was chosen due to its conceptual simplicity yet implementation of core concepts of quantum mechanics. It provides a solid foundation for entering the realm of understanding quantum computing.

The two main objectives are:

1. Produce a 2D graph to represent the evolution of measurement probability for a specific quantum state over time
2. Plot a quantum state evolving on a 3D Bloch Sphere

4.1.1 Expected outcomes

Bloch Sphere

By understanding vector space axioms and matrix multiplication (section 1.1.2.1, 1.2.1) we were able to run make a solid prediction on how the Pauli-X matrix will affect the quantum state $|0\rangle$ in Section 2.5.1. Now we will outline the desired / expected results of how all three Pauli matrices will affect the 3 principal bases. The table below was produced via the working out in section 4.1.1.1.

Quantum state/ Pauli Gates	X	Y	Z
$\|0\rangle$	$\|1\rangle$	$i\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$	$-i\|1\rangle$	$-\|1\rangle$
$\|+\rangle$	$\|+\rangle$	$-i\|-\rangle$	$\|-\rangle$
$\|-\rangle$	$-\|-\rangle$	$i\|+\rangle$	$\|+\rangle$
$\|i\rangle$	$i\|-i\rangle$	$\|i\rangle$	$\|-i\rangle$
$\|-i\rangle$	$-i\|i\rangle$	$-\|-i\rangle$	$\|i\rangle$

- RED = Global Phase
- Green = Bit flip
- white = no change

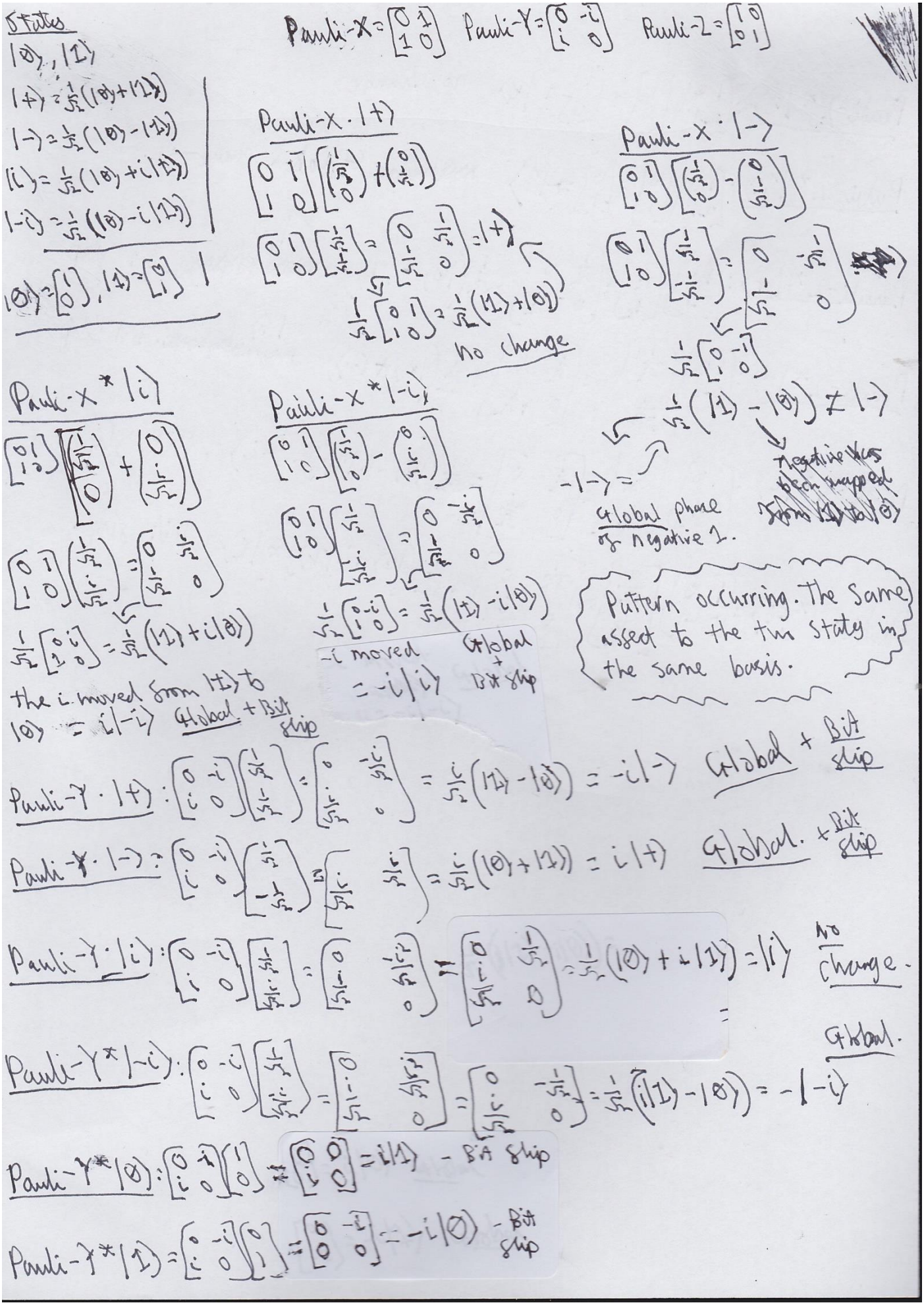
2D Graphs

These are the expected effects of each Pauli matrix on different quantum states, which we can now compare with simulation results.

Based on the transformations each matrix applies; we predict the following Rabi oscillations when used in Hermitian Hamiltonians:

1. Pauli-X should induce oscillations between Z-basis ($|0\rangle$,$|1\rangle$) and Y-basis ($|i\rangle$,$|-i\rangle$), due to its bit-flip nature.
2. Pauli-Y is expected to drive oscillations between X-basis ($|+\rangle$,$|-\rangle$) and Y-basis ($|0\rangle$,$|1\rangle$) combining bit- and phase-flip effects.
3. Pauli-Z should cause oscillations between both X-basis ($|+\rangle$, $|-\rangle$) and Y-basis ($|i\rangle$,$|-i\rangle$), as it introduces relative phase changes.

4.1.1.1 Prediction workings out



Pauli-Z * $|0\rangle$: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$ no change

Pauli-Z * $|1\rangle$: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -|1\rangle$ ~~no change~~ Global phase -1.

Pauli-Z * $|+\rangle$: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) = |-\rangle$ ~~no change~~. Bit flip

Pauli-Z * $|-\rangle$: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & +\frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) = |+\rangle$ ~~no change~~. Bit flip

Pauli-Z * $|i\rangle$: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{i}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}\left(|0\rangle - i|1\rangle\right) = |-i\rangle$ = Bit flip

Pauli-Z * $|-i\rangle$: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{i}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{i}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}\left(|0\rangle + i|1\rangle\right) = |i\rangle$ = Bit flip.

4.2 Code structure and logic Development

This section will be brief, as the complexity of the project arises primarily from the underlying mathematics and quantum mechanics that drive the functions being implemented.

The text in the red boxes highlights fixes or lessons learned during the development process.

The purple-highlighted boxes show that the bold, purple coloured code lines are part of the preceding code block.

4.2.1 Imports and Setup:

Importing only the essential libraries to ensure sufficient functionality while maintaining optimal computation speed and efficiency. Each for the reasons mentioned in Chapter 3.

```
import numpy as np
import matplotlib.pyplot as plt
from qutip import *
```

This code block is named : Library Import 1

4.2.2 Initial Conditions

Unless stated otherwise, all values provided in the 'Initial Conditions' section should be considered as the default values used.

4.2.2.1 Quantum States

We can initialise the computation standard basis $|0\rangle$ and $|1\rangle$, then build the other 2 principal bases using them. Using the Qutip Basis function:

Init0 = basis(2, 0) = $|0\rangle$

Init1 = basis(2, 1) = $|1\rangle$

(see section 3.2.1 on how basis define quantum states)

Using Init0 and Init1 to build other quantum states

i_state = (Init0 + 1j * Init1).unit() = $|i\rangle$

neg_i_state= (Init0 - 1j * Init1).unit() = $|-i\rangle$

plus_state = (Init0 + Init1).unit() = $|+\rangle$

neg_state= (Init0 - Init1).unit() = $|-\rangle$

This code block is named : Quantum States 1

.unit()

The .unit() is a qutip function to normalize the state. i_state = (Init0 + 1j * Init1) itself this creates the state

$|\psi\rangle = |0\rangle + i|1\rangle$

if we calculate its inner product (section 1.1.4), it does not equal 1. However, (Init0 + 1j * Init1).unit() = $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, which will allow for measurements to be accurate. So, all quantum states , save the Z-basis, will require the '.unit()' addition.

4.2.2.2 Rabi Frequency and Time points Array

For the Rabi frequency, we will use **Ω** (uppercase Greek letter omega) to represent the value.

Ω = np.pi // using the numpy library for the accurate pi value (Section 3.3.2)

The array for the time points is initialized using another numpy method 'linspace' detailed in Section 3.4

Tlist = np.linspace(0, 5, 500)

This creates an array of 500 evenly spaced time points between 0 and 5, which represent the time steps at which the simulation will compute and return values to be used in the model.

This code block is named : Rab + Time values 1

4.2.3 Hamiltonians

The Pauli operators used in this section (introduced in Section 2.5.1) are already available as built-in functions in QuTiP, as discussed in Section 3.3. We now construct the Hermitian operators (as introduced in Section 2.5) using these Pauli matrices and the Rabi frequency (Ω), which characterizes the strength of the driving field.

HX = Ω * sigmax() // Driving Hamiltonian: H = (Ω) * σ_x

HZ = Ω * sigmaz() // Driving Hamiltonian: H = (Ω) * σ_z

HY = Ω * sigmay() // Driving Hamiltonian: H = (Ω) * σ_y

If the Rabi frequency Ω is varied, the strength of the driving term in the Hamiltonian will change accordingly.

This code block is named : Hamiltonians 1

4.2.3.1 Observables

As started in section 2.6.3, there are two types of observables (Projectors and Pauli operators) and are built differently (Section 3.5).

Projectors

Excited_Proj_ init1 = Init1 * Init1 dag()

Creating the project from a previously initialized quantum state Init1. Setting as the excited state (See section 2.6.3.1 for more on excited/ground states).

The build of the observable will be made clear in the name, for example above, it was made using quantum state init1. If one was made using init0, it would be titled Excited_Proj_init0.

Pauli operators

As stated in 3.5.2, simply using the inbuilt Pauli operators in the position of observables will provide us Pauli Expectation Values.

4.2.1 Calculating results

There are two main ways we will calculate data using the mesolve function , with and without observables. Using the Syntax in section 4.3 , we can insert the previously initialized values.

result = mesolve(H, Initial, Time_points, [], O)

H ∈ {HX, HZ, HY}

Initial ∈ {Init0, Init1, i_state, neg_i_state, plus_state, neg_state}

Time_points = Tlist

O ∈ {Excited_Proj_init1, sigmax(), sigmaz(). sigmay() }

> O = must always be an array type and surrounded via square brackets [] when passed to the mesolve function, even when a single value.

The data will be labelled according to the Hamiltonian driving the evolution. For example, if H = HX, result = HX_evolution.

4.2.2 Producing graphs

Using the syntax stated in section 3.6, we insert the results produced by the mesolve function in 4.2.4. **In addition to the Code in 3.6.1 and 3.6.2**

Both 2D graphs and the Bloch Spheres require the results.expect[i] method mentioned in 3.5.

2D Graph example

```
…

result = mesolve(H, Initial, Tim_points, [], O)

...

plt.plot(tlist, result.expect[0], label="P(|1⟩)")
```

Code Block for the creation of 2D Graphs : 2D Graph 1

Bloch Sphere example

```
…

Result1 = mesolve(H, Initial, Tim_points, [], )

Result2 = mesolve(H, Initial, Tim_points, [], O)

Result3  = mesolve(H, Initial, Tim_points, [], O)

...

b_x.add_points([Result1.expect[0], Result2.expect[0], Result3.expect[0]])

…
```

Code Block for the creation of Bloch Spheres : Block Sphere 1

4.3 Plotting Results on Bloch Sphere

4.3.1 Bloch Sphere 6-single Qubits

To begin with, I began plotting the 6 quantum states on a Bloch sphere at time 0 , to see where they began. The code was built as :

- Library Import 1
- Rabi + Time Values 1
- Hamiltonians 1
- Quantum States 1
- Excited_State_Init1

For efficiency I used a for loop using the length of an array that contained all quantum state values.

QuantumState_List = [Init0, Init1,i_state,neg_i_state,plus_state,neg_state]

for i in range (0,len(QuantumState_List)):

Code Block of the for loop for all 6 quantum states: Simulate All States 1

- Three mesolve functions HY_evolution, HZ_evolution, HX_evolution using:
  - QuantumState_List[i] for the Initial value
  - corresponding Hamiltonian drivers and Pauli operators as observables
  - Time array initialized earlier

(For values on the Bloch sphere, we must provide three values due to this being a 3D graph , see section 2.7 and 3.6.2)

- Block Sphere 1

To add the data points to the Bloch sphere, the syntax used was:

b_x.add_points([HX_evolution.expect[0][0], HY_evolution.expect[0][0],.HZ_evolution.expect[0][0])

The reason for using expect[0][0] is that expect[0] returns a list of expectation values over time for the observable, and [0] accesses the **first-time step**. So, expect[0][0] gives the expectation value of the observable at **time t = 0**.

4.3.2 Bloch Sphere Hamiltonian-X driven Rabi Oscillation

The next program should be to see the evolution of the qubits on the Bloch sphere driven by a Hamiltonian using the Pauli-X Gate. The Code is as follows:

- Library Import 1
- Rabi + Time Values 1
- Hamiltonians 1
- Quantum States 1

We can reuse the for loop used in 4.3.1:

QuantumState_List = [Init0, Init1,i_state,neg_i_state,plus_state,neg_state]

for i in range (0,len(QuantumState_List)):

  - HX_evolution with no observable, (like the syntax found in 3.4). Using Time array initialized earlier and QuantumState_List[i] for the Initial value.
  - X,Y,Z Vector components at each time point calculated (see red box underneath)
  - **Block Sphere 1**

- HX_x_vals = [expect(sigmax(), state) for state in HX_evolution.states]
- HX_y_vals = [expect(sigmay(), state) for state in HX_evolution.states]
- HX_z_vals = [expect(sigmaz(), state) for state in HX_evolution.states]

These lines manually calculate the Bloch vector components for each time step in the evolution.

- HX_evolution.states is a list of quantum states (Qobj) at each time point in the Time array.
- expect(operator, state) calculates the expectation value ⟨operator⟩ for a given state.
- The list comprehension loops through each time-evolved state to calculate the corresponding expectation value for σ_x, σ_y, and σ_z.

Providing you the x,y,z values of an evolving qubit which can be provided the to Bloch sphere.

Bloch.add_points([HX_x_vals, HX_y_vals, HX_z_vals])

This program produces the following results:

Figure 4.3.2 i_state

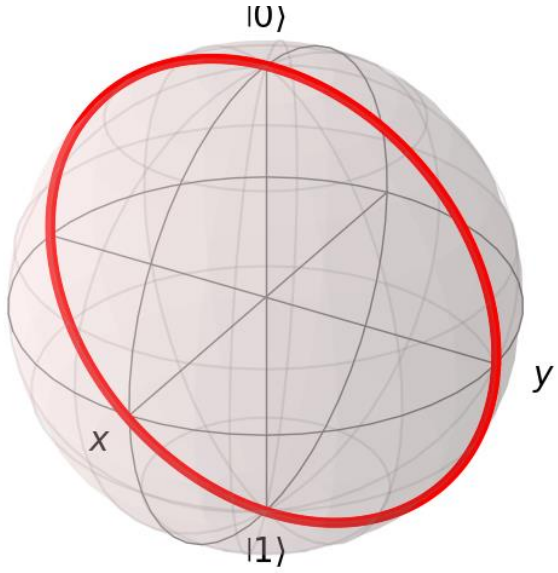


Figure 4.3.2 neg_i_state

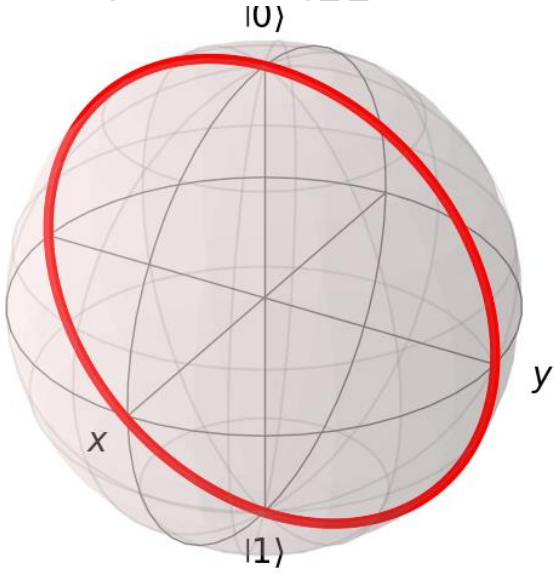


Figure 4.3.2 neg_state

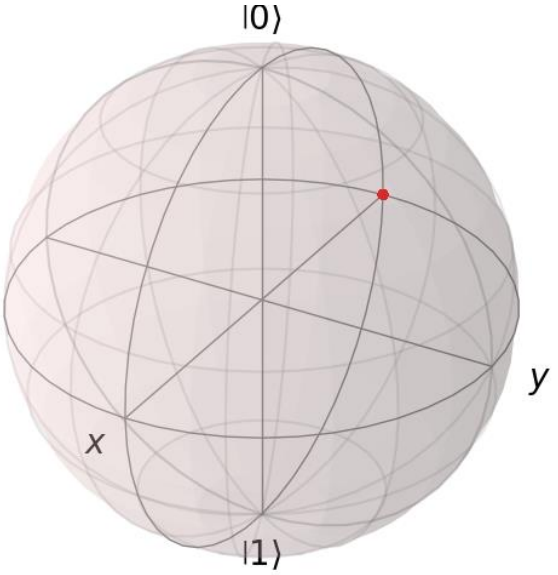


Figure 4.3.2 plus_state

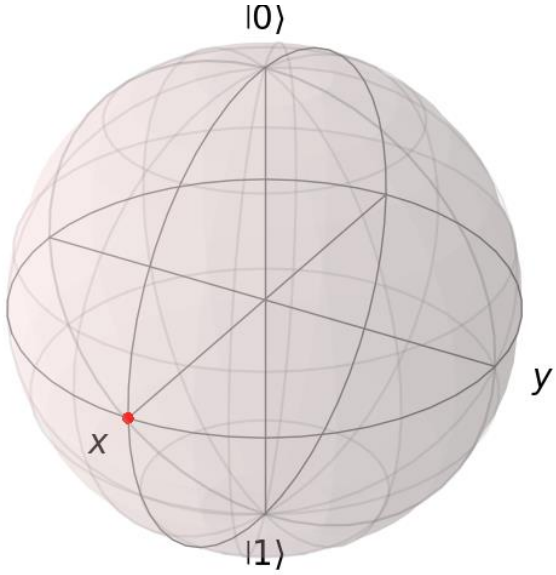


Figure 4.3.2 one_state

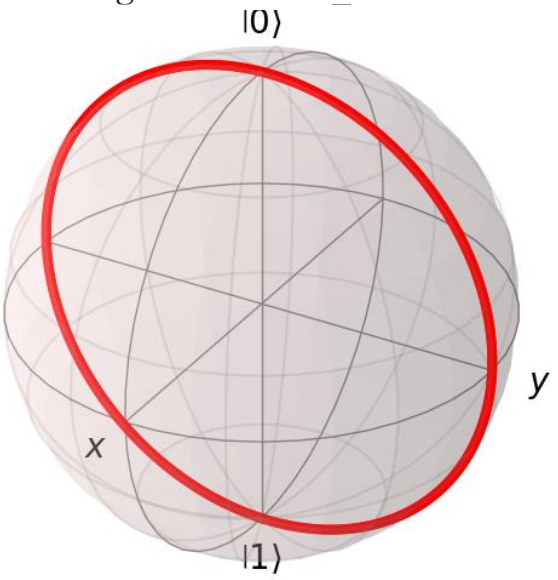
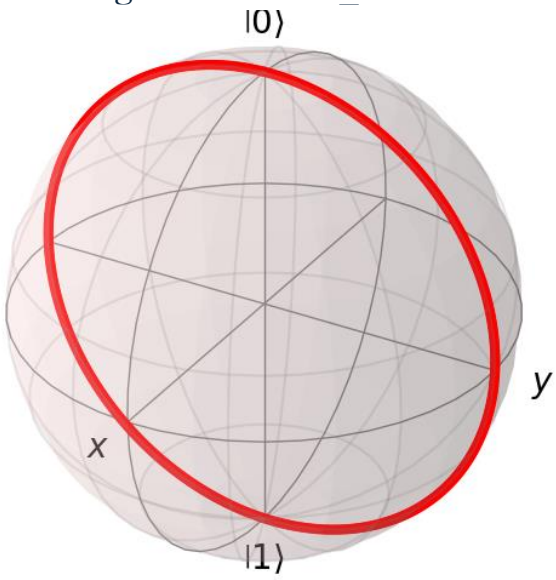


Figure 4.3.2 zero_state



4.3.3 Bloch Sphere Hamiltonian-Y driven Rabi Oscillation

This program will be built exactly like 4.3.2 except the Pauli-Y operator will be utilized to drive the qubit evolution.

HX_evolution turns to HY_evolution

This program produces the following results:

Figure 4.3.3 i_state

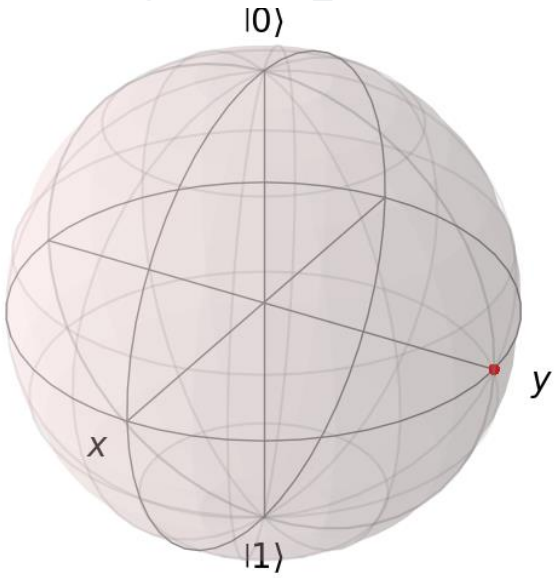


Figure 4.3.3 neg_i_state

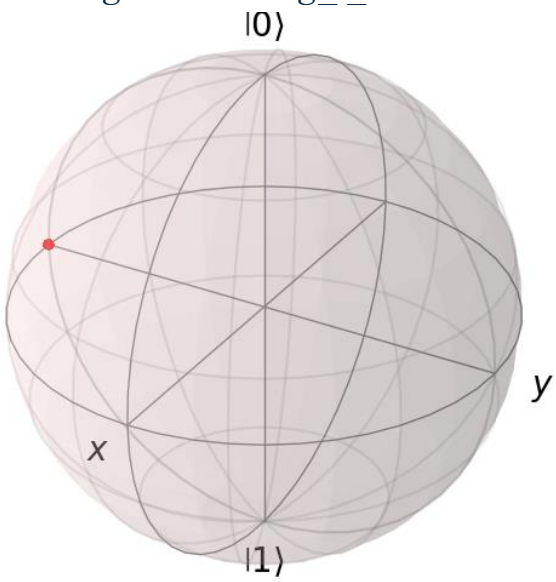


Figure 4.3.3 neg_state

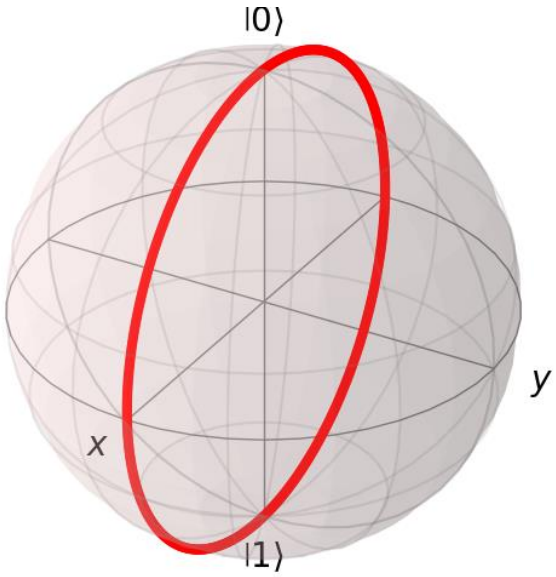


Figure 4.3.3 plus_state

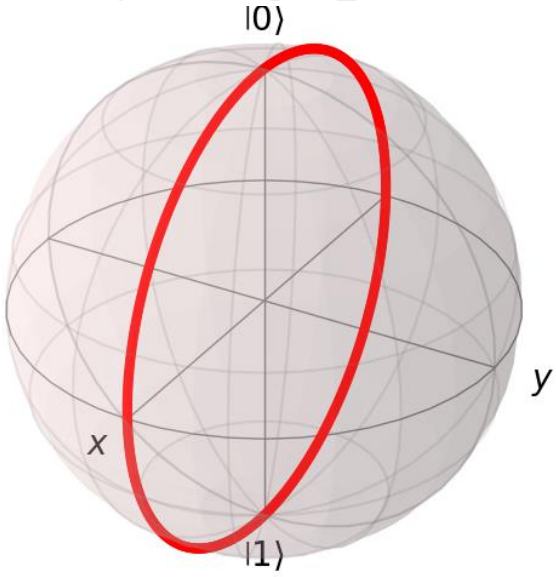


Figure 4.3.3 one_state

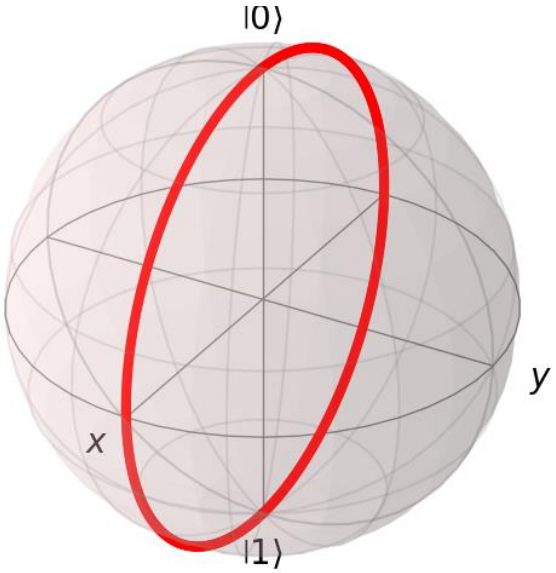
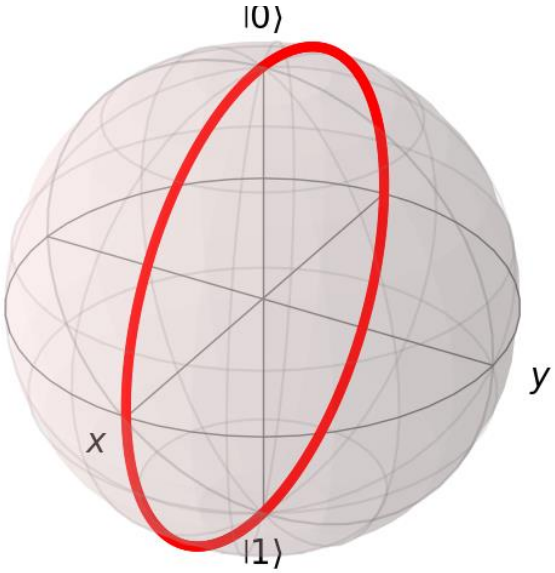


Figure 4.3.3 zero_state



4.3.4 Bloch Sphere Hamiltonian-Z driven Rabi Oscillation

This program will be built exactly like 4.3.2 except the Pauli-Z operator will be utilized to drive the qubit evolution.

HX_evolution turns to HZ_evolution

This is program produces the following results:

Figure 4.3.4 i_state

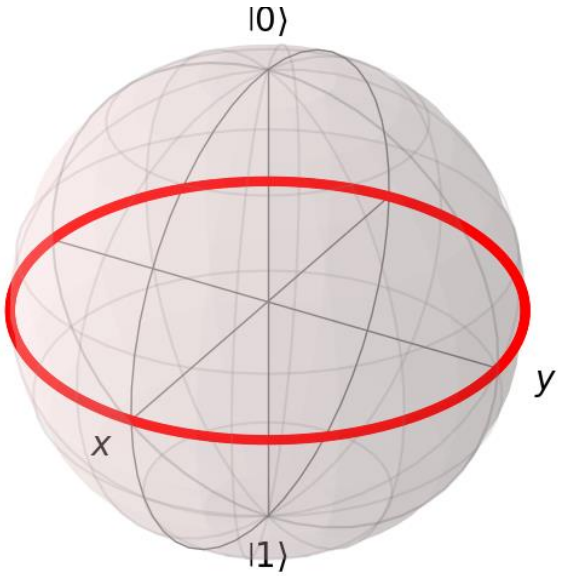


Figure 4.3.4 neg_i_state

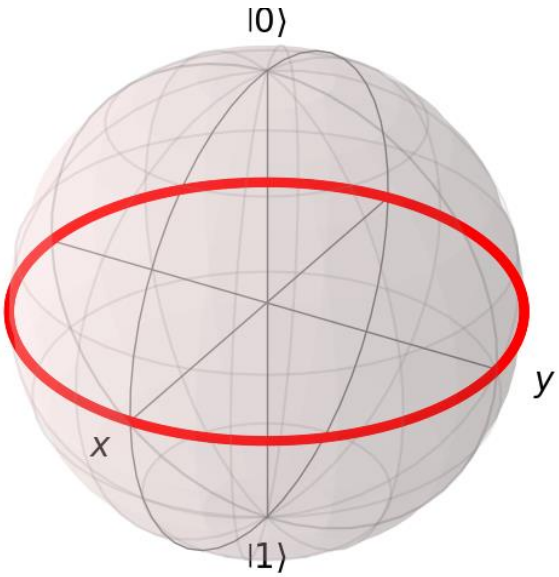


Figure 4.3.4 plus_state

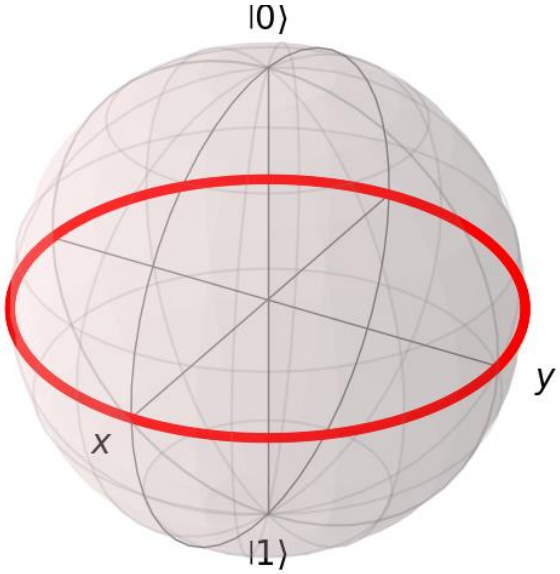


Figure 4.3.4 neg_state

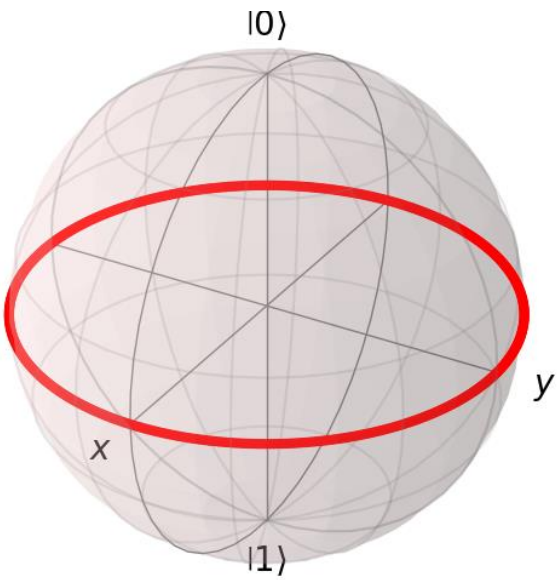


Figure 4.3.4 one_state

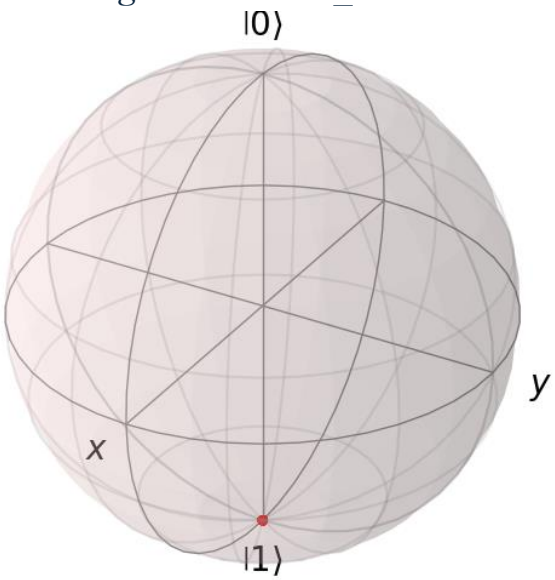
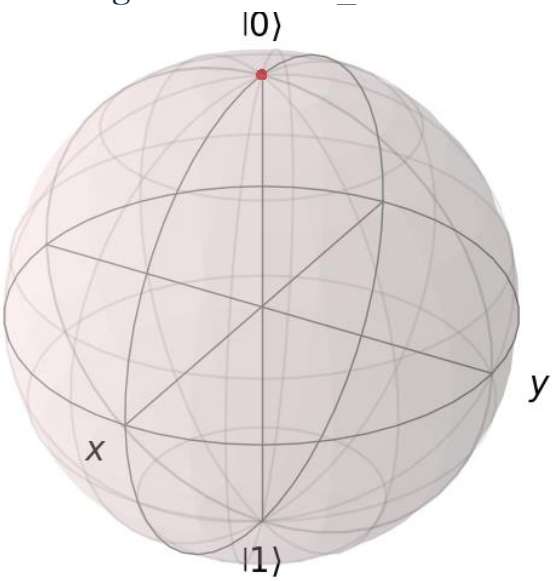


Figure 4.3.4 zero_state



4.4 Plotting Results on 2D Graph

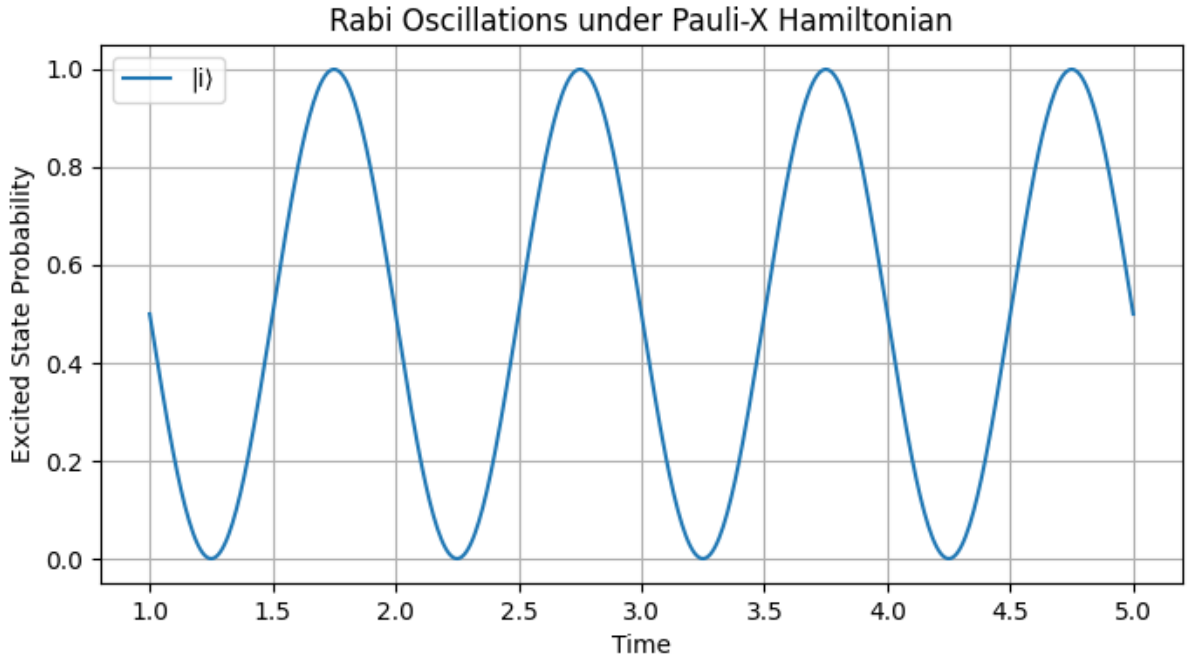
4.4.1 Probability of Excited State init1

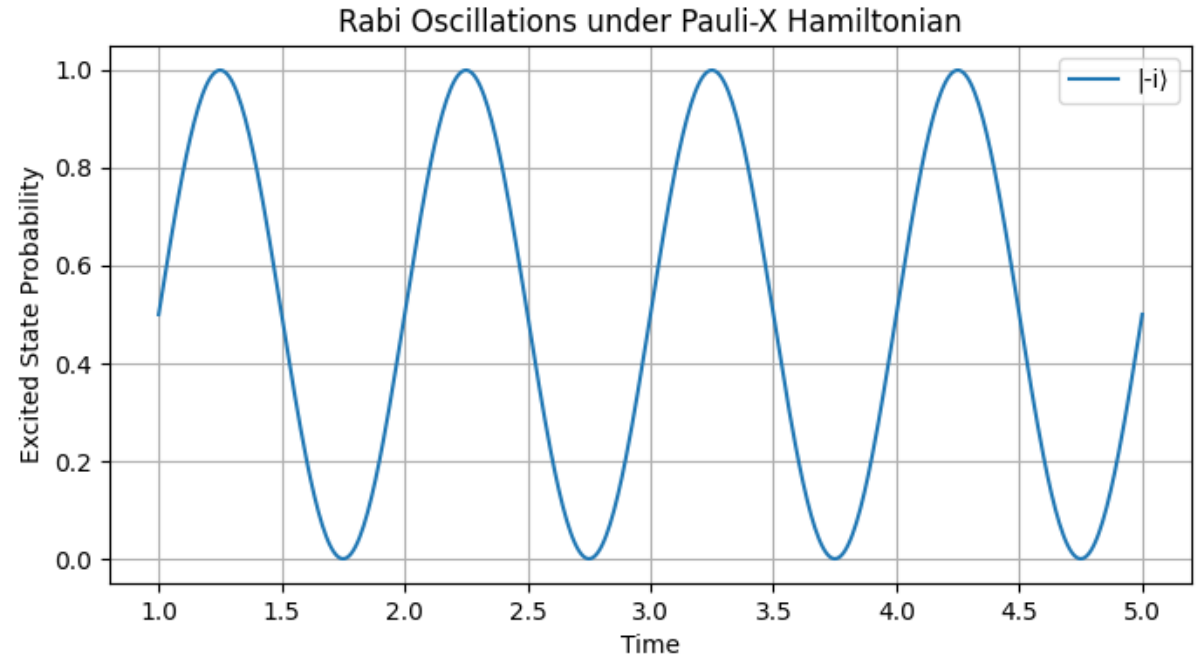
Next, we create 2D graphs using expectation values (Section 2.6.3) to visualize the probability of measuring the quantum system in a specific state. These probabilities are calculated based on the initial qubit values defined in Section 4.2.2.1. The code is as follows:

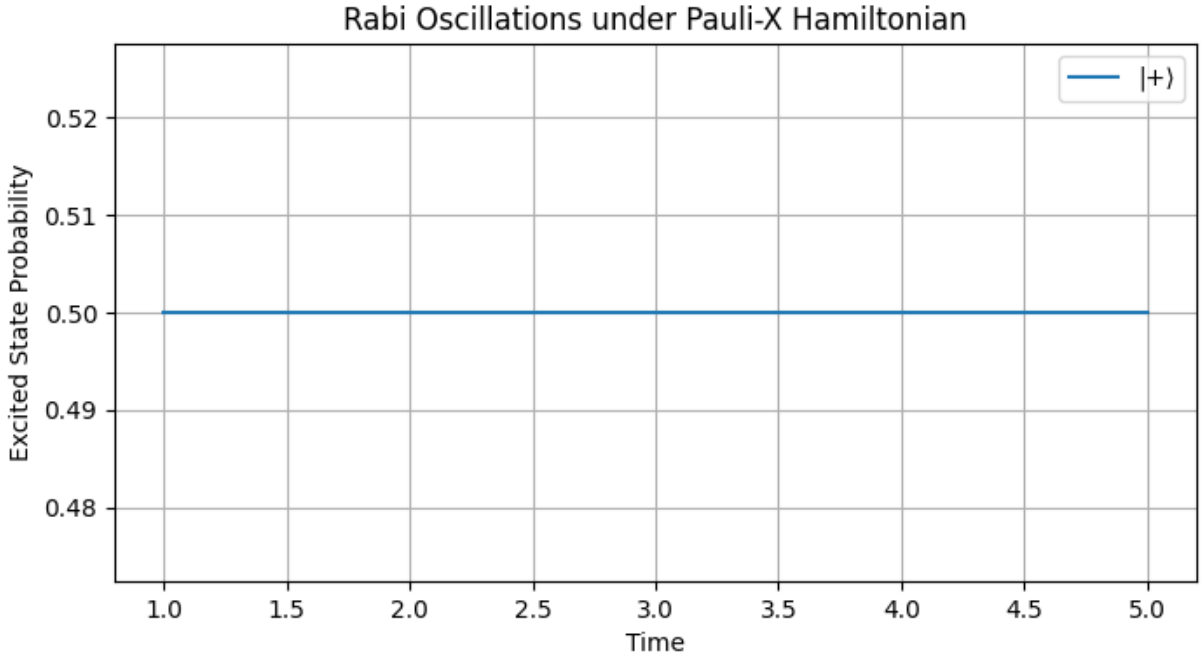
- Library Import 1
- Quantum States 1
- Rabi + Time Values 1
- The Pauli-X Hemertian operator from Hamiltonians 1
- Excited_Proj_init1
- For Loop for 0 to [array of all hemertian operators [HX,HY,HYZ]]
    - Simulate All States 1
        - Mesolve with
            - Hamiltonian = Hamiltonians[U],
            - Initial = QuantumState_List[i],
            - Time point array = Tlist,
            - Observable = Excited_Proj_init1
            - 2D Graph 1

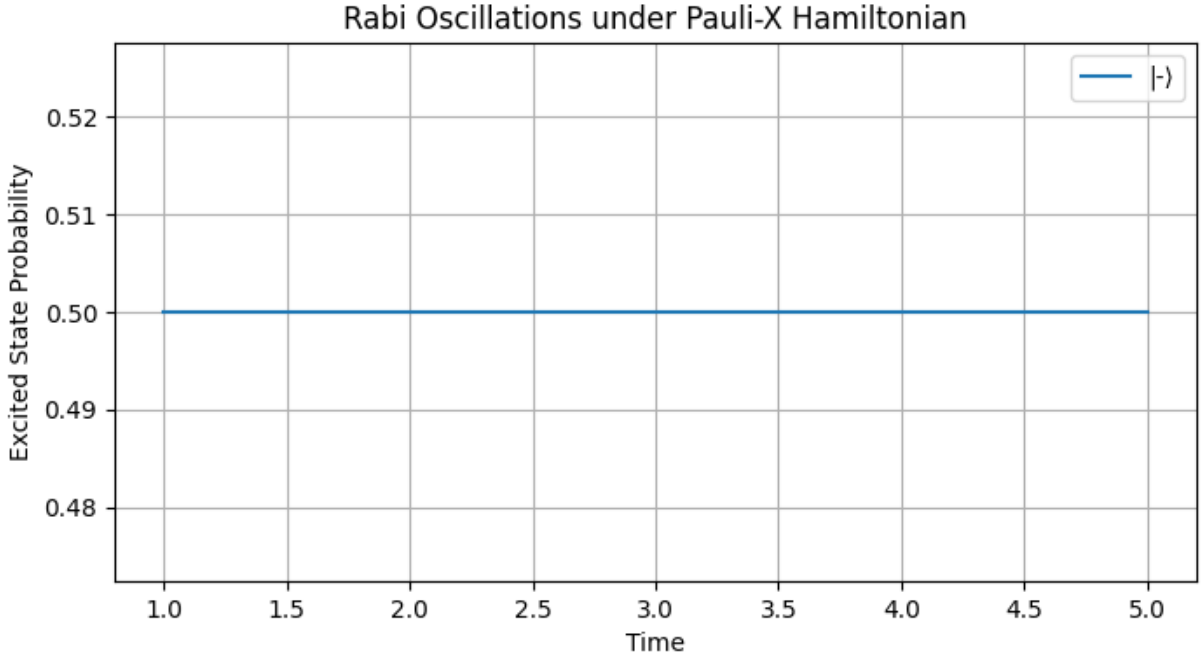
The line plt.show() must be set outside the for loops for all graphs to generate and appear at the same time, instead of 1 at a time

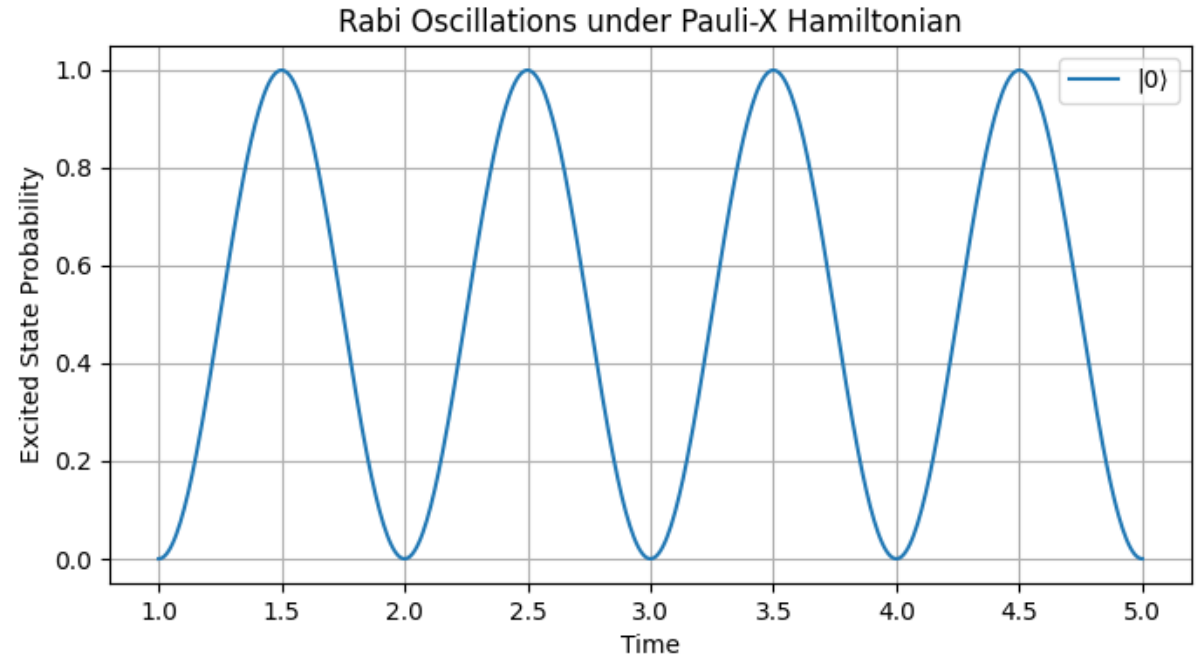
This program generates a series of results in which the initial qubit state and the Pauli operator used in the Hamiltonian vary, while the excited state—whose probability is being measured—remains fixed as $|1\rangle$.

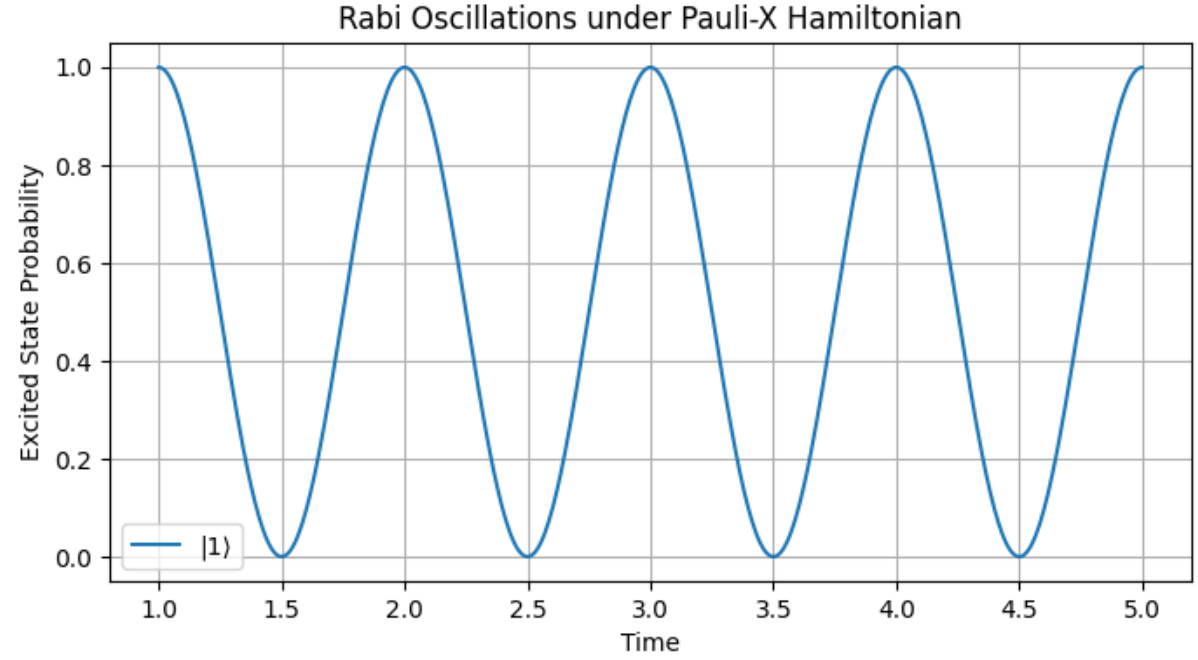


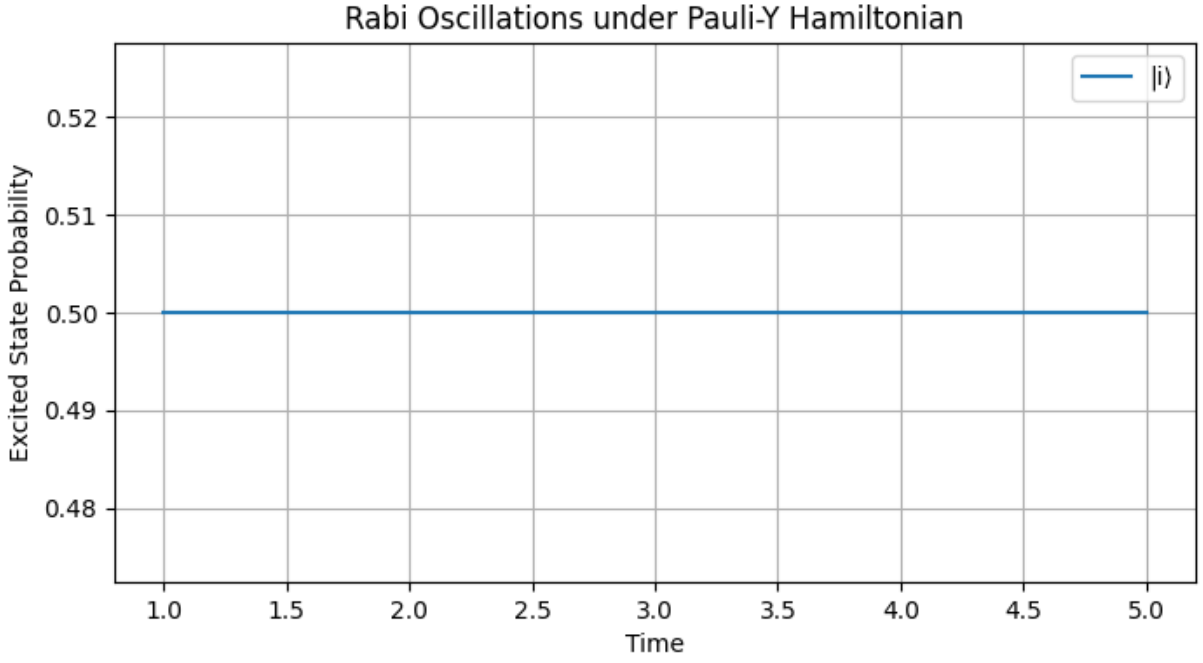

Rabi Oscillations under Pauli-X Hamiltonian
|-i⟩
Excited State Probability
Time
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

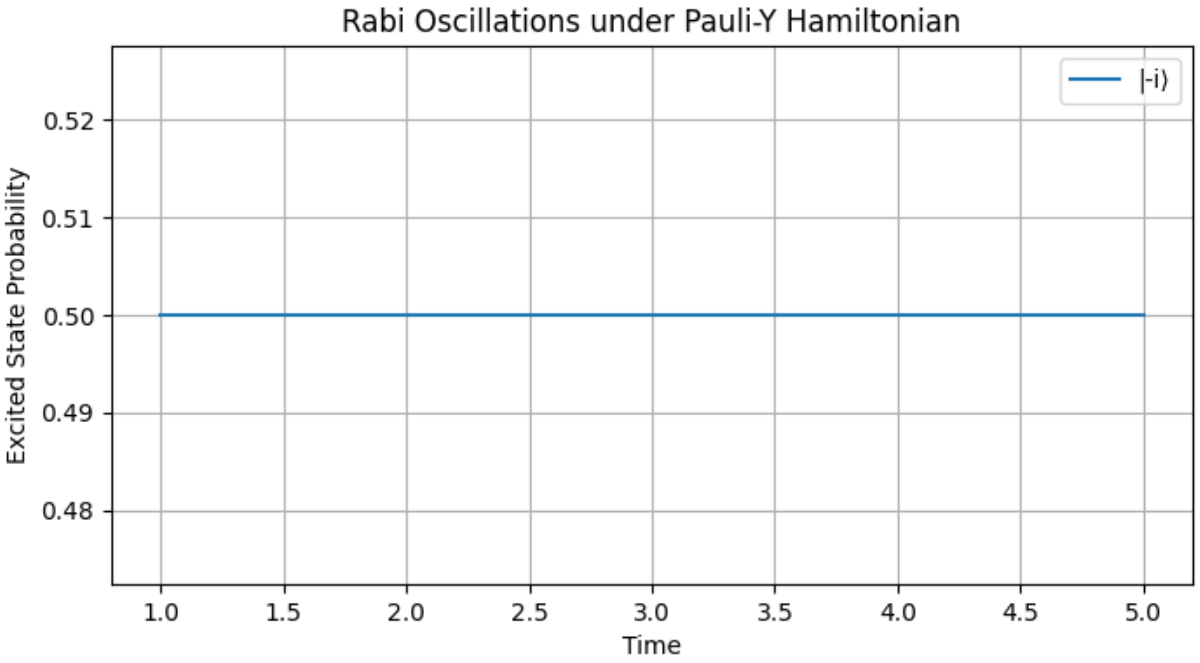

Rabi Oscillations under Pauli-X Hamiltonian
|+⟩
Excited State Probability
Time
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

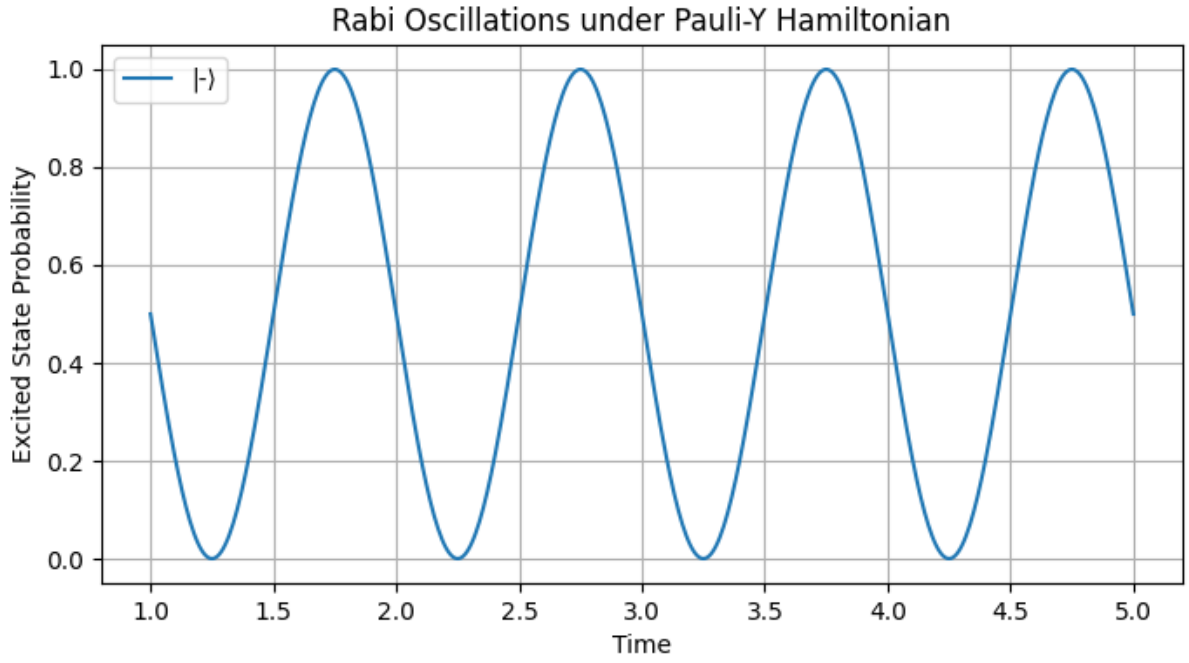

Rabi Oscillations under Pauli-X Hamiltonian
|-⟩
Excited State Probability
Time
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

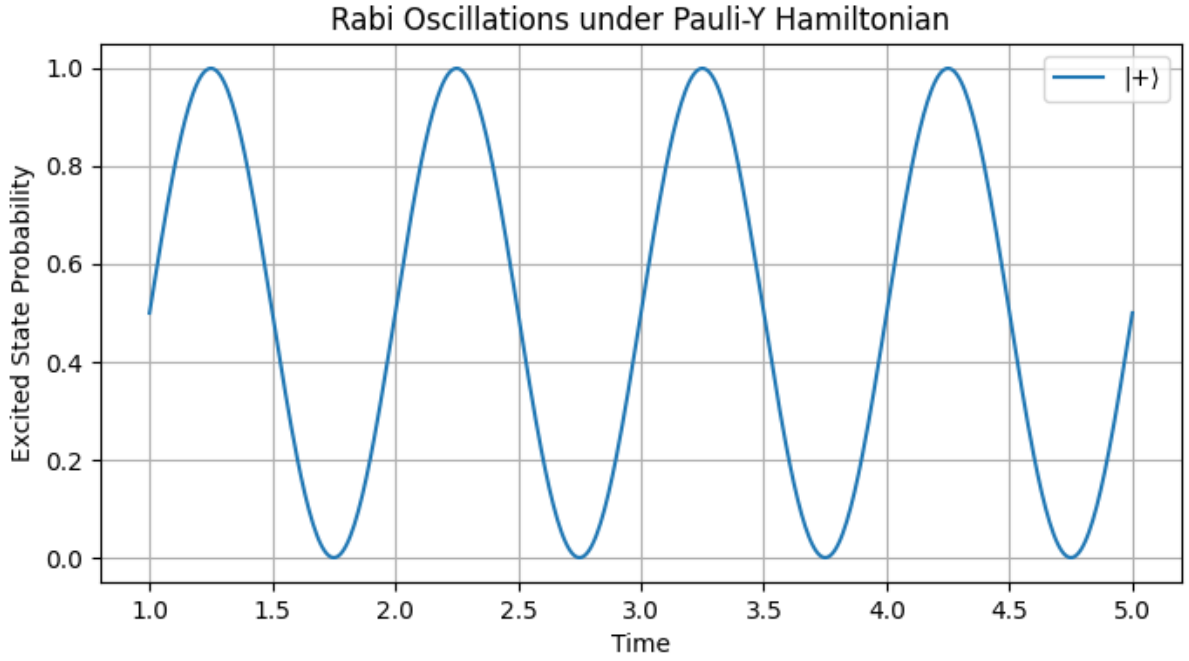

Rabi Oscillations under Pauli-X Hamiltonian
Excited State Probability
Time
|0⟩
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

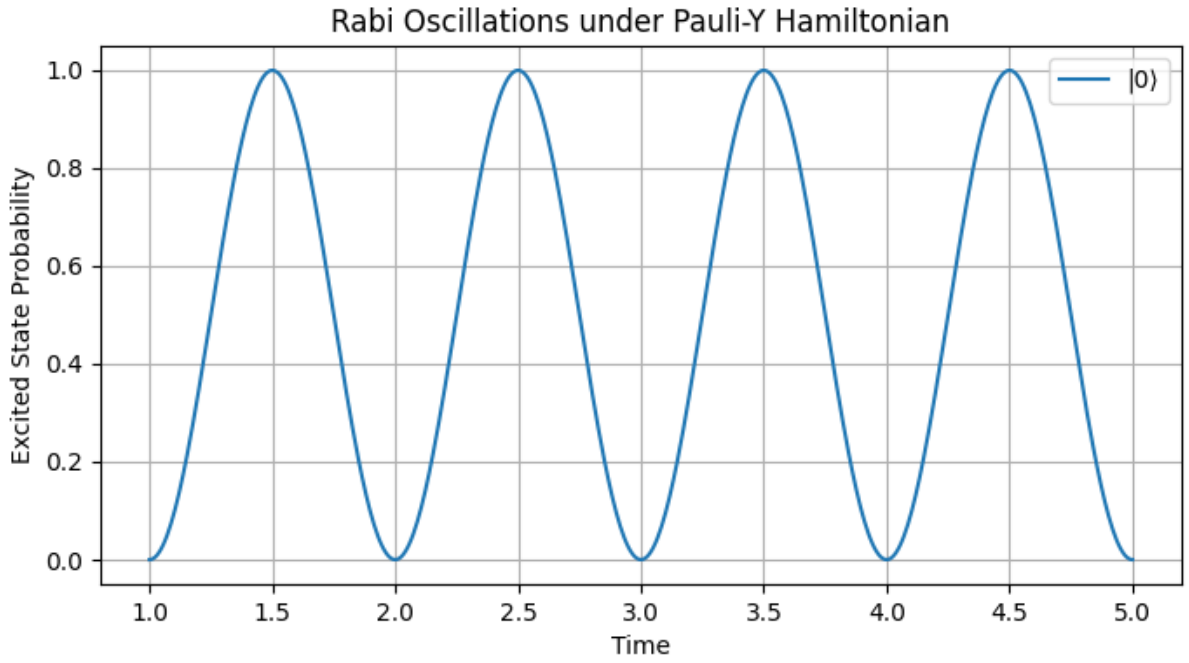

Rabi Oscillations under Pauli-X Hamiltonian
Excited State Probability
Time
|1⟩
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

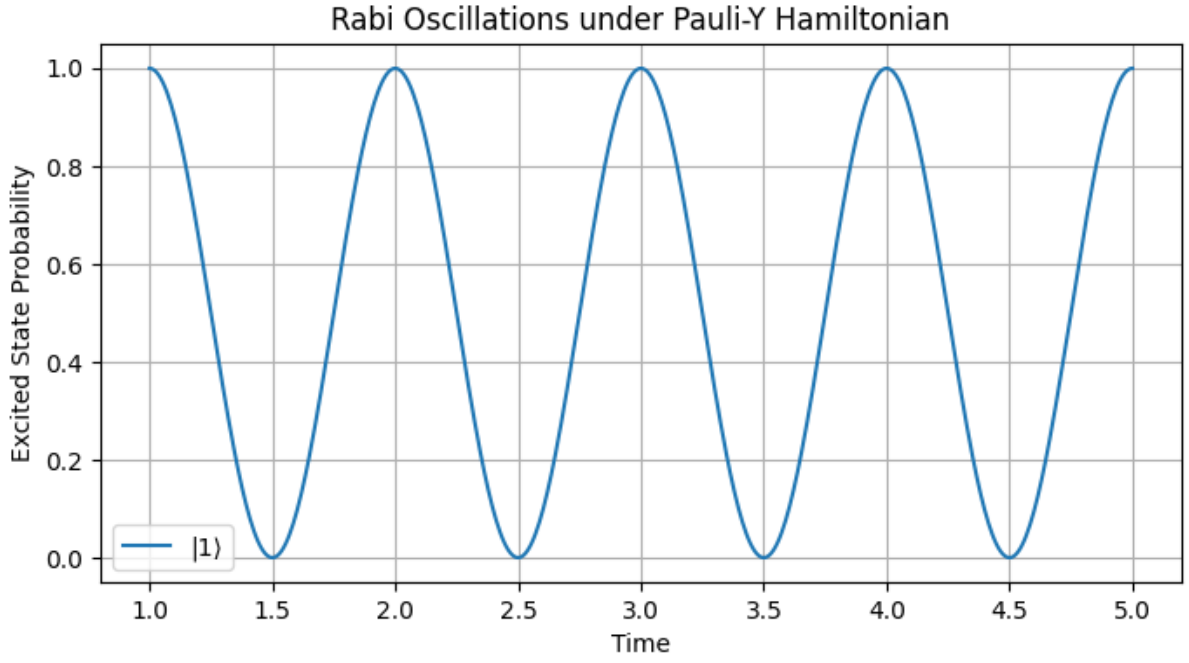

Rabi Oscillations under Pauli-Y Hamiltonian
Excited State Probability
Time
|i⟩
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

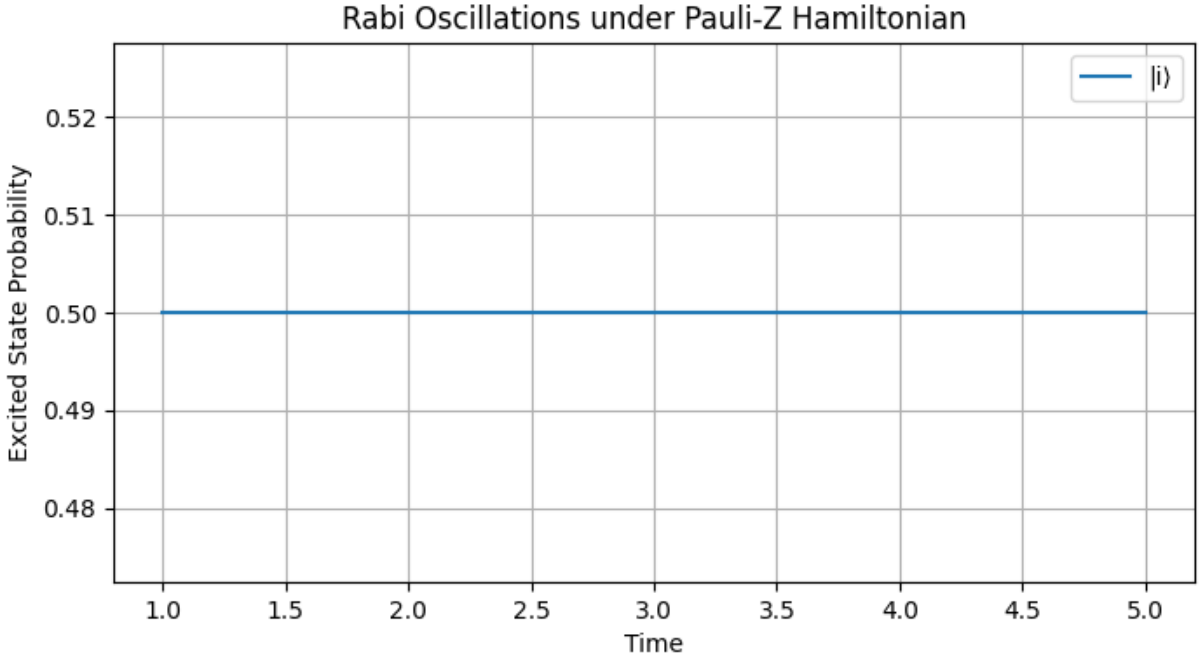

Rabi Oscillations under Pauli-Y Hamiltonian
|-i⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

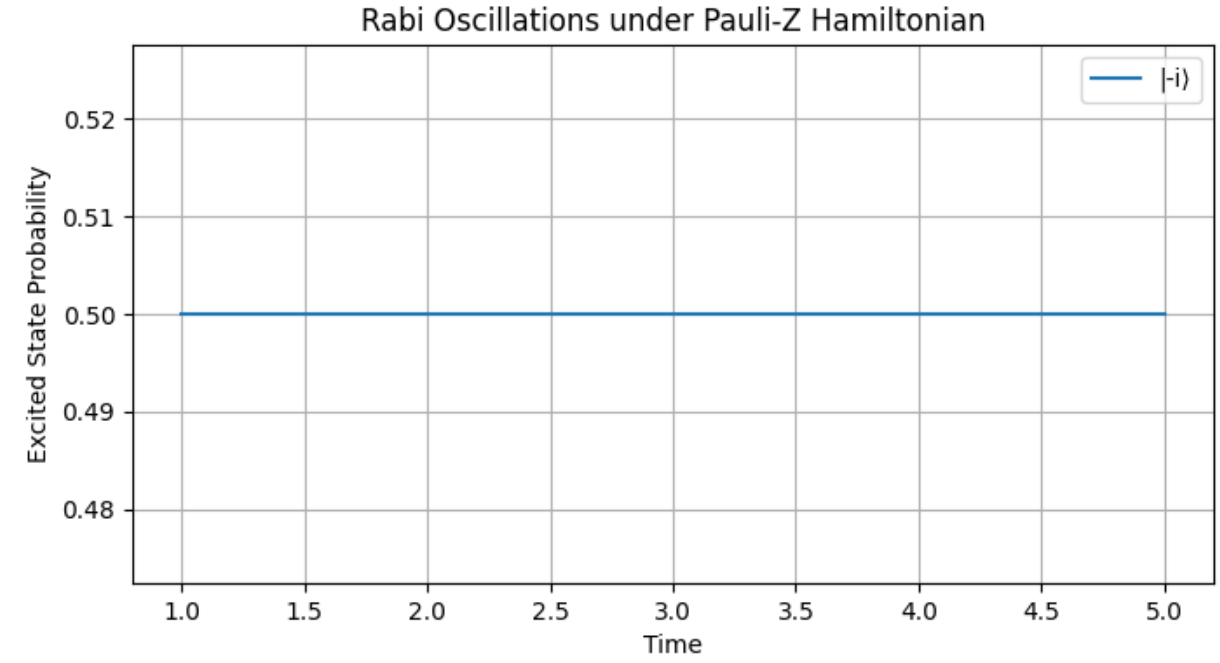

Rabi Oscillations under Pauli-Y Hamiltonian
|-⟩
Excited State Probability
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

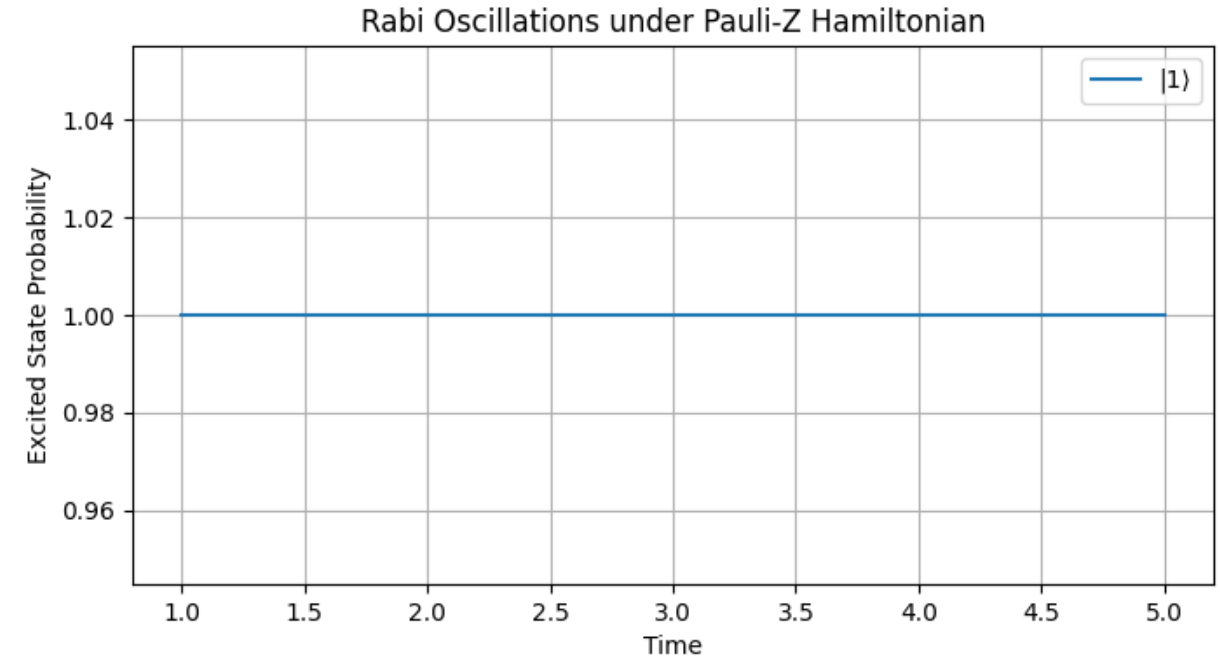

Rabi Oscillations under Pauli-Y Hamiltonian
|+⟩
Excited State Probability
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

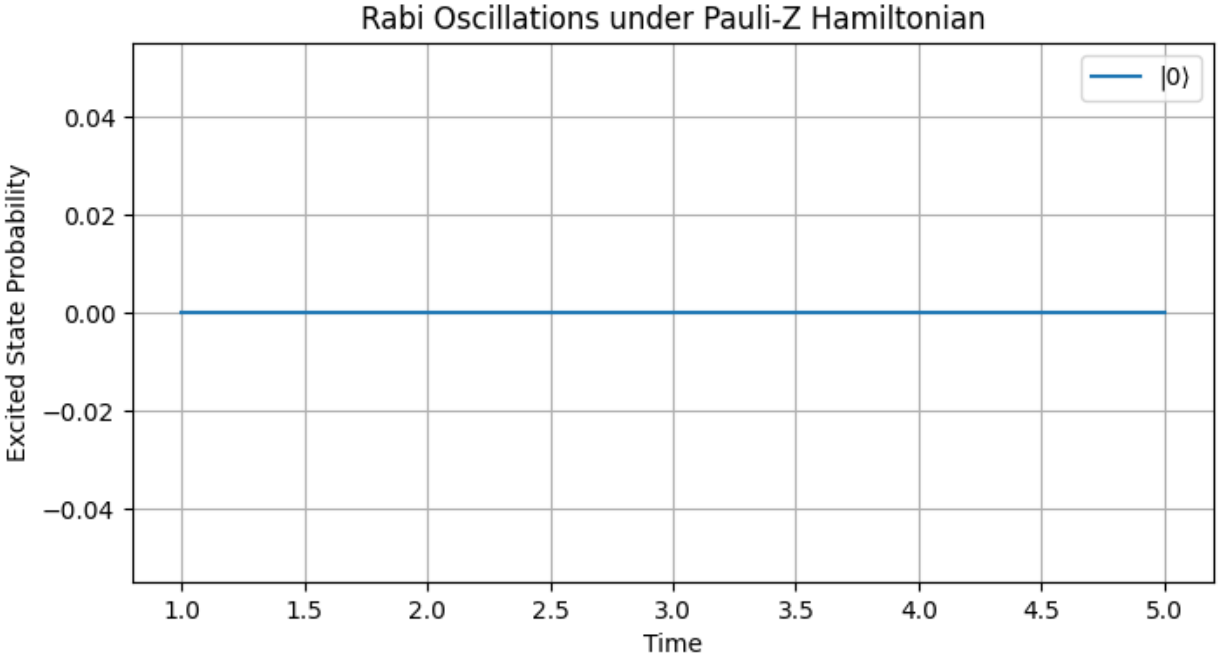

Rabi Oscillations under Pauli-Y Hamiltonian
Excited State Probability
Time
|0⟩
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

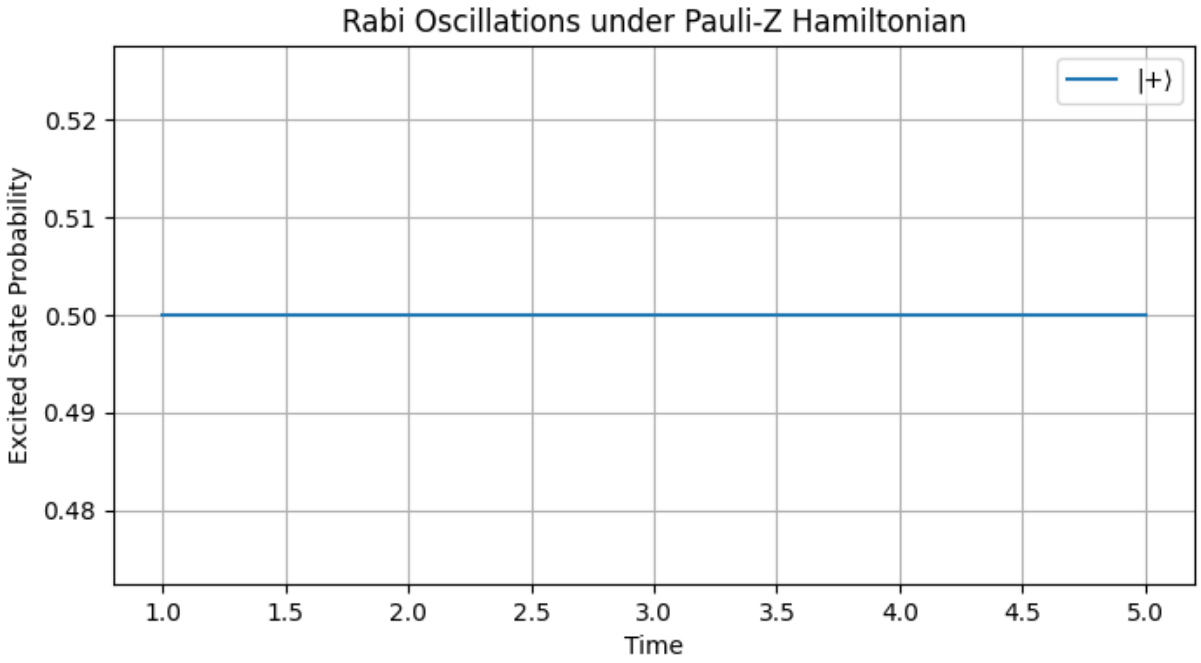

Rabi Oscillations under Pauli-Y Hamiltonian
Excited State Probability
Time
|1⟩
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

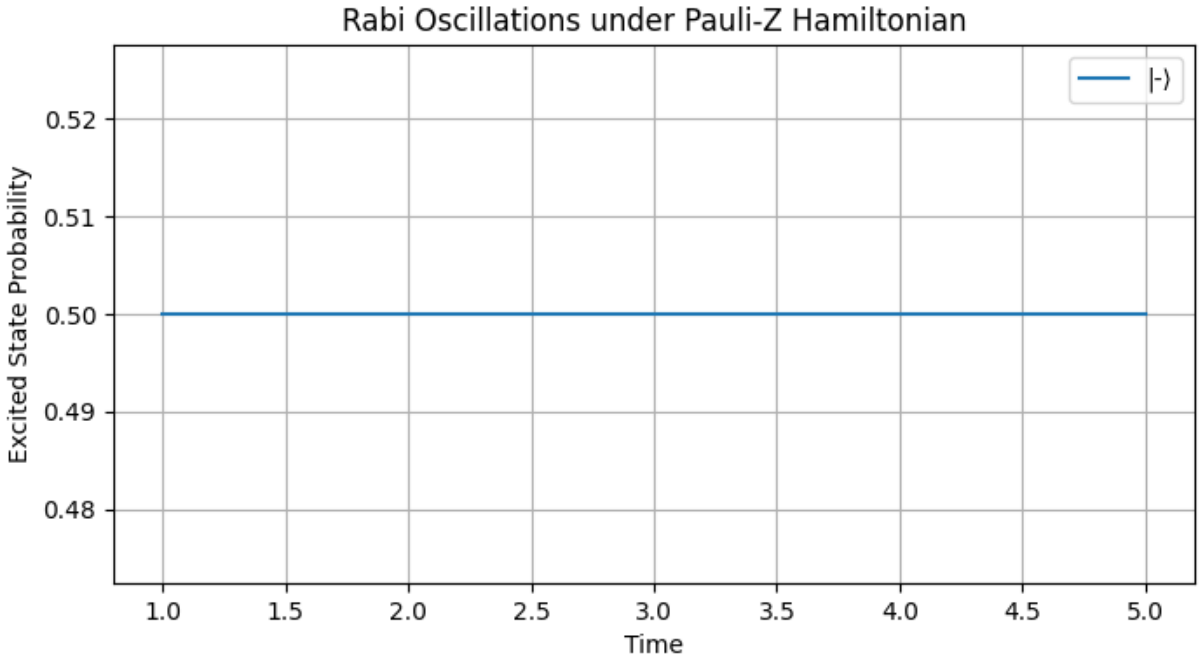

Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|i⟩
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Rabi Oscillations under Pauli-Z Hamiltonian
|-i⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time


Rabi Oscillations under Pauli-Z Hamiltonian
|1⟩
Excited State Probability
1.04
1.02
1.00
0.98
0.96
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time


Rabi Oscillations under Pauli-Z Hamiltonian
|0⟩
Excited State Probability
0.04
0.02
0.00
−0.02
−0.04
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

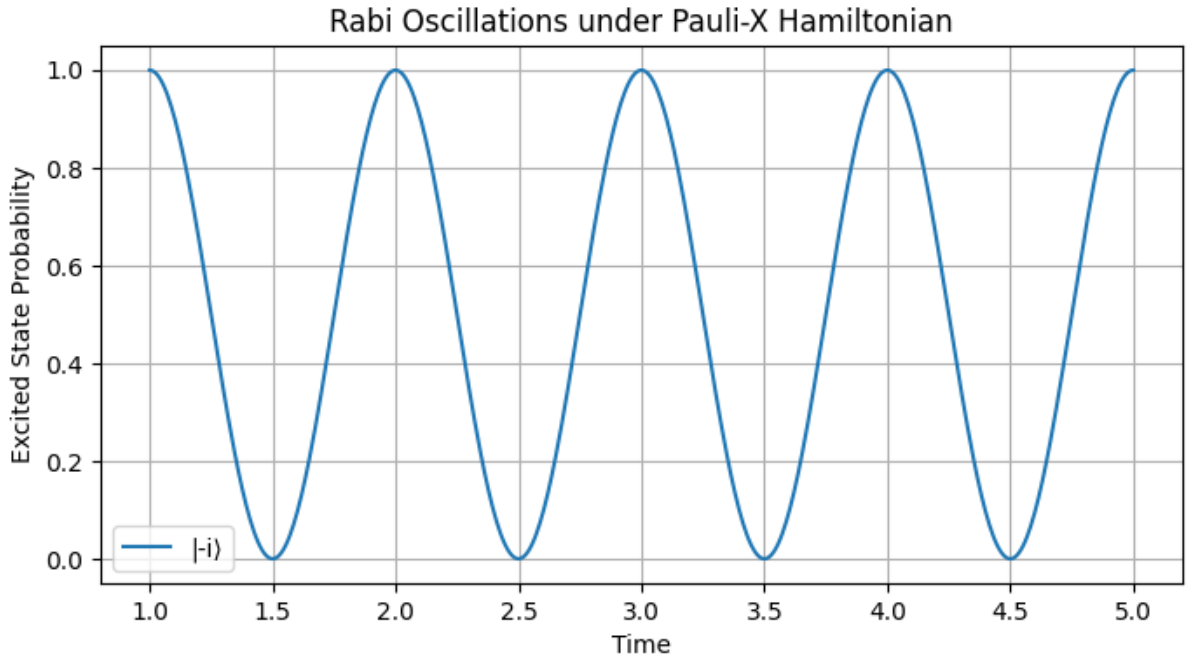


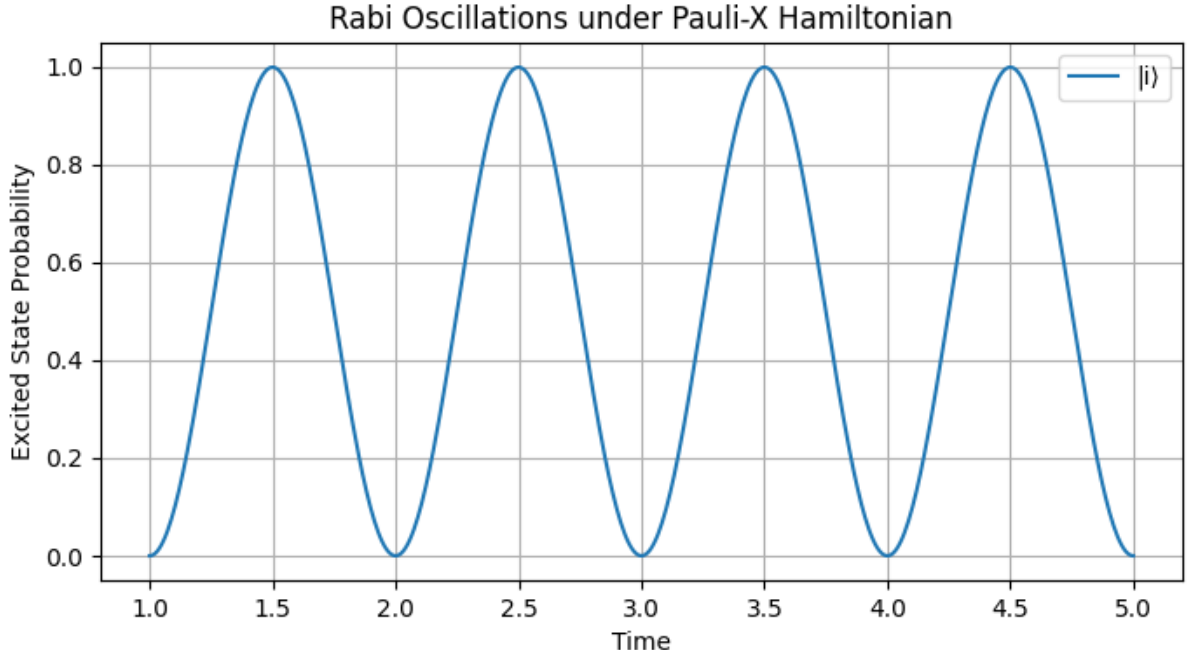


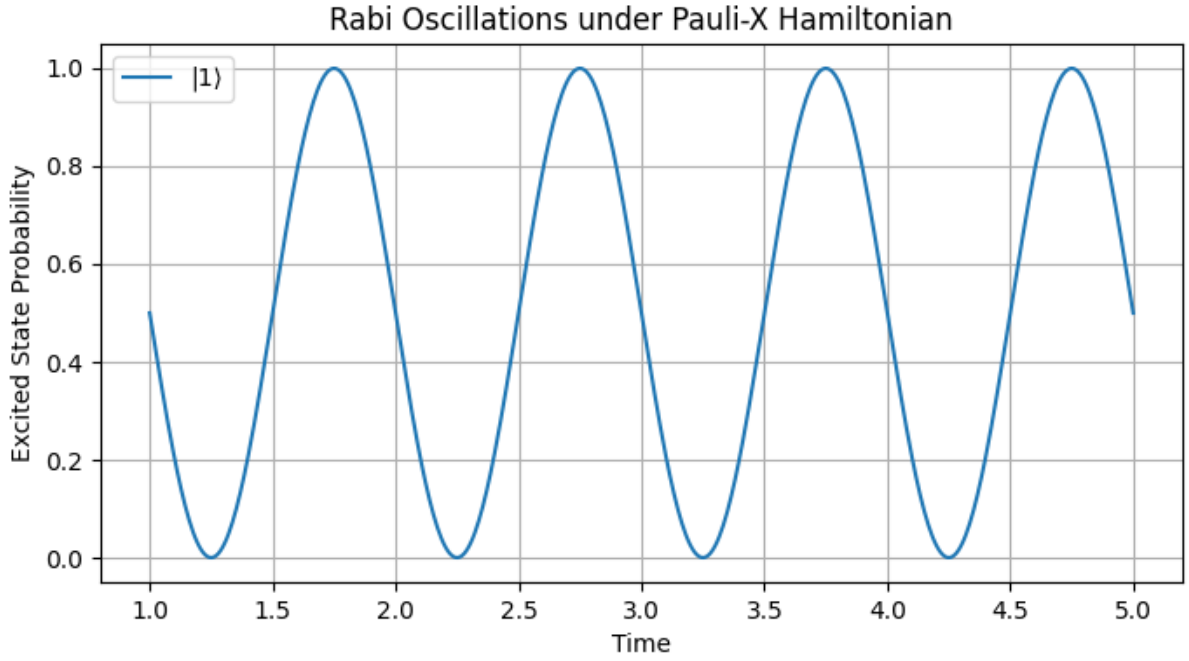
4.4.2 Probability of Excited State neg_i_state

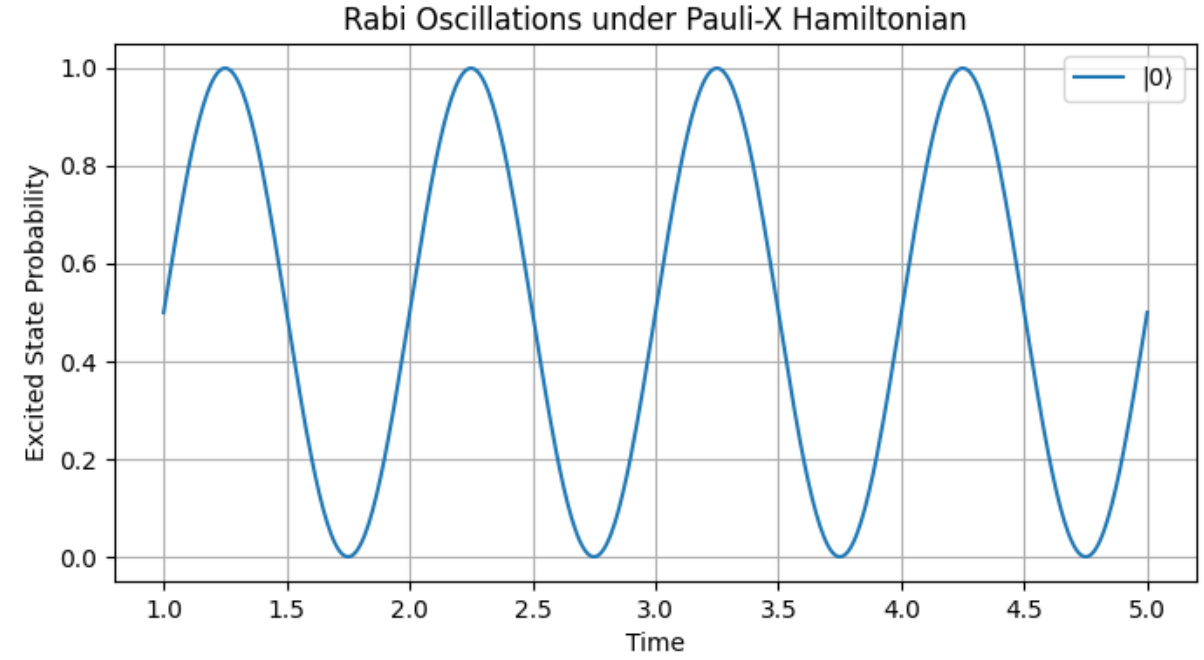
This program is exactly as outlined in 4.4.1, except Excited_Proj_init1 is swapped for neg_i_state to produce results measuring the probability of being in state

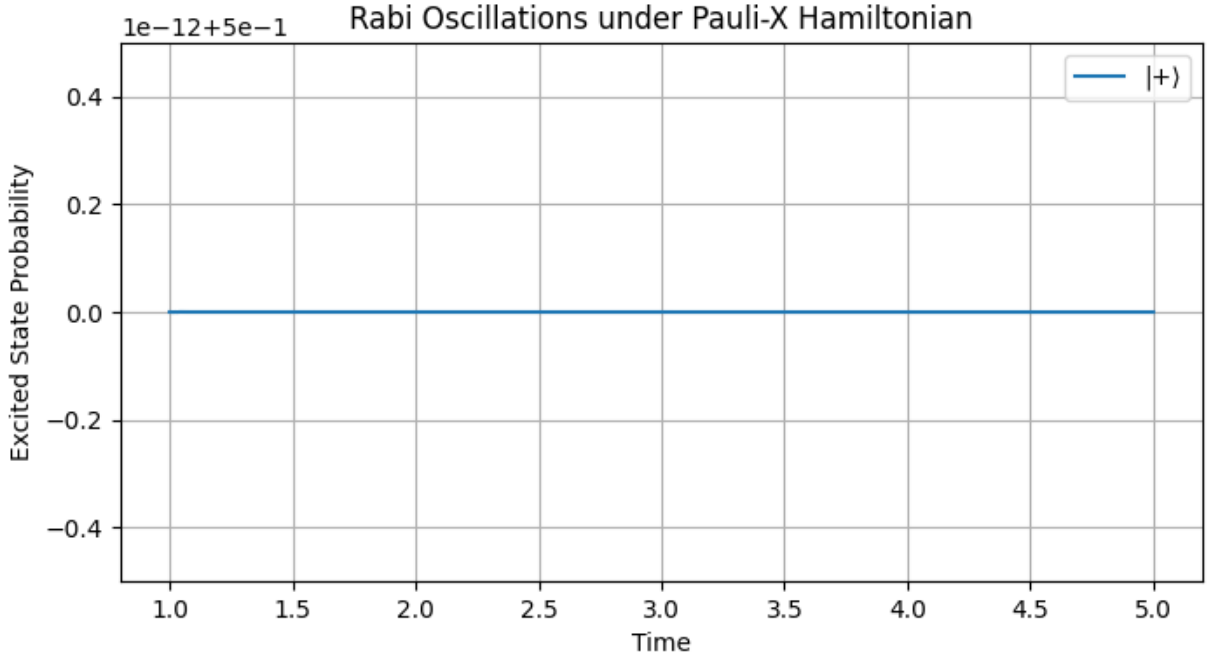
This program produces the following results.

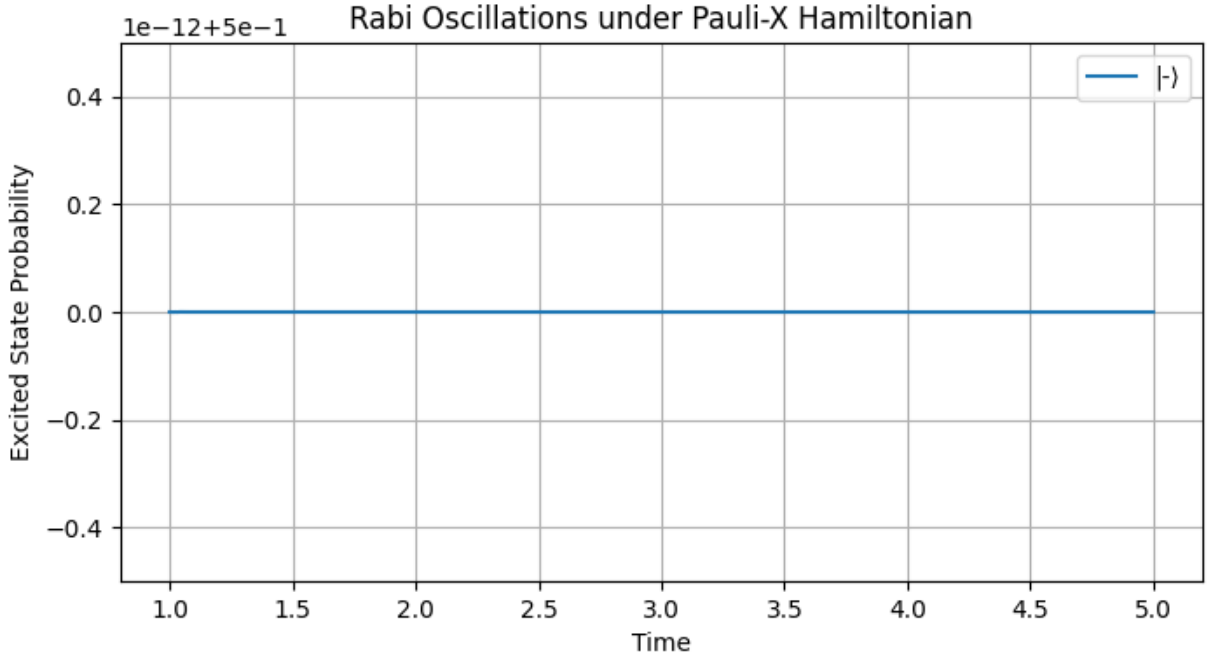

Rabi Oscillations under Pauli-X Hamiltonian
Excited State Probability
Time
|-i⟩
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

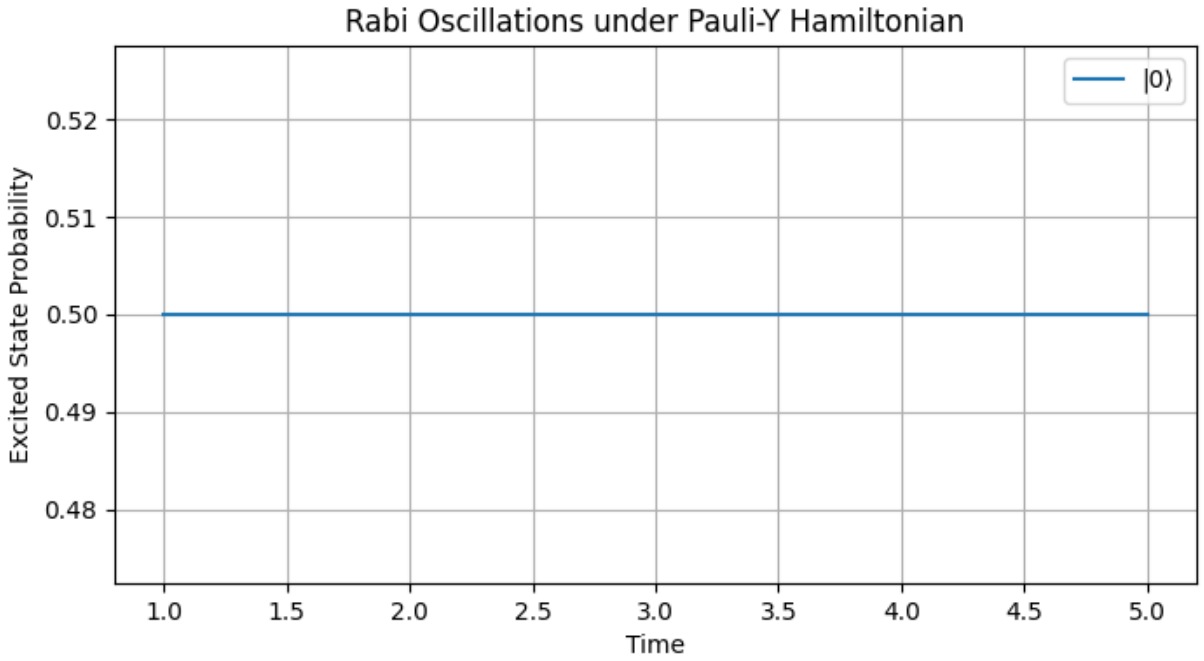

Rabi Oscillations under Pauli-X Hamiltonian
Excited State Probability
Time
|i⟩
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

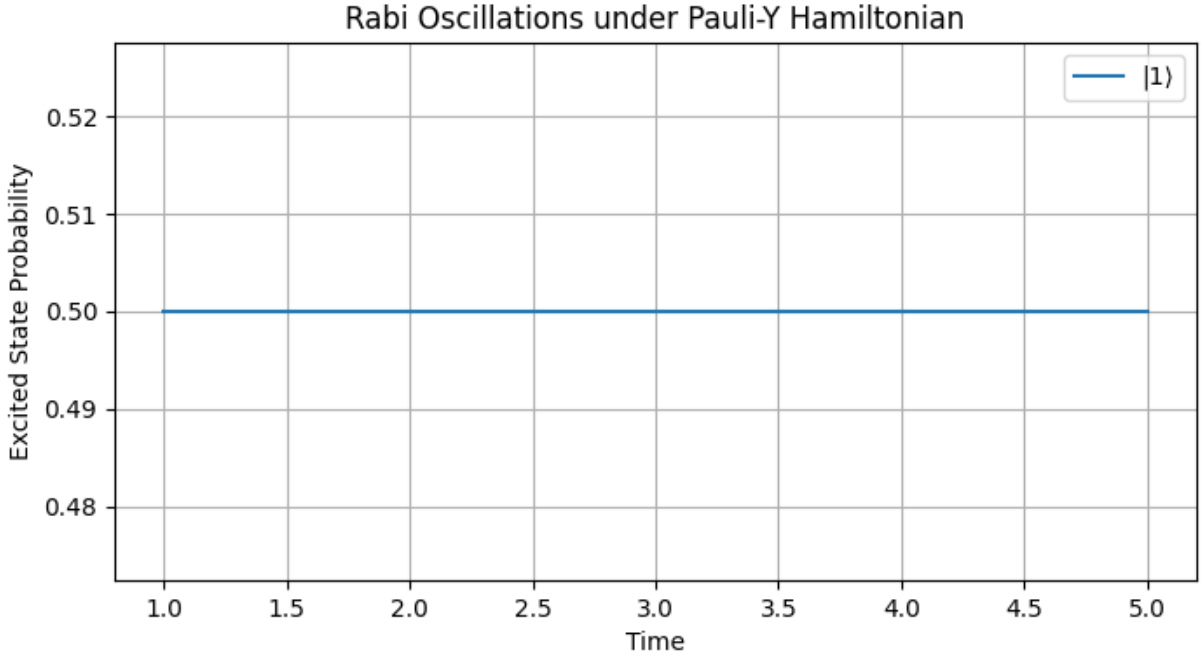

Rabi Oscillations under Pauli-X Hamiltonian
Excited State Probability
Time
|1⟩
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

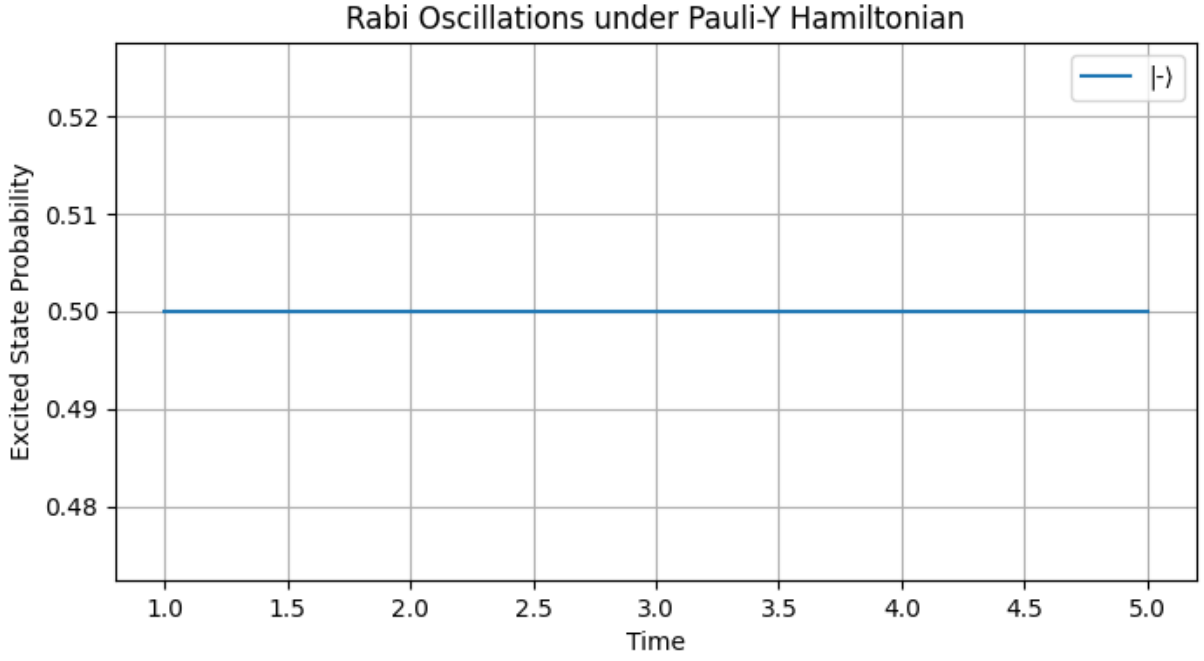

Rabi Oscillations under Pauli-X Hamiltonian
|0⟩
Excited State Probability
Time

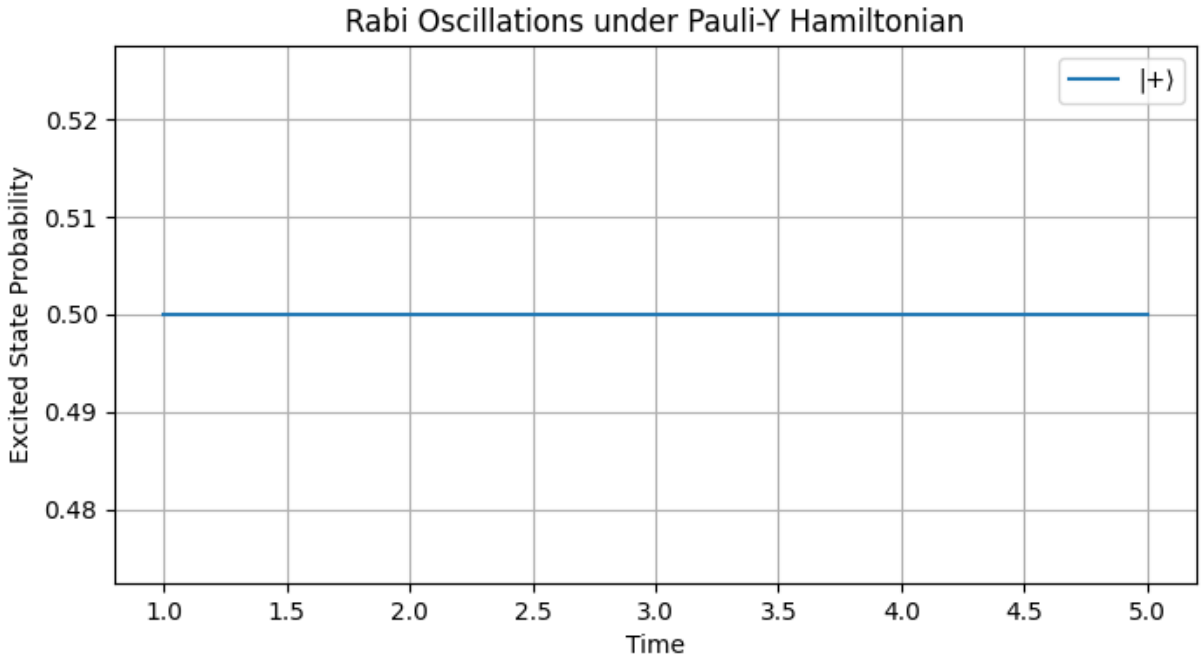

Rabi Oscillations under Pauli-X Hamiltonian
1e−12+5e−1
|+⟩
Excited State Probability
Time

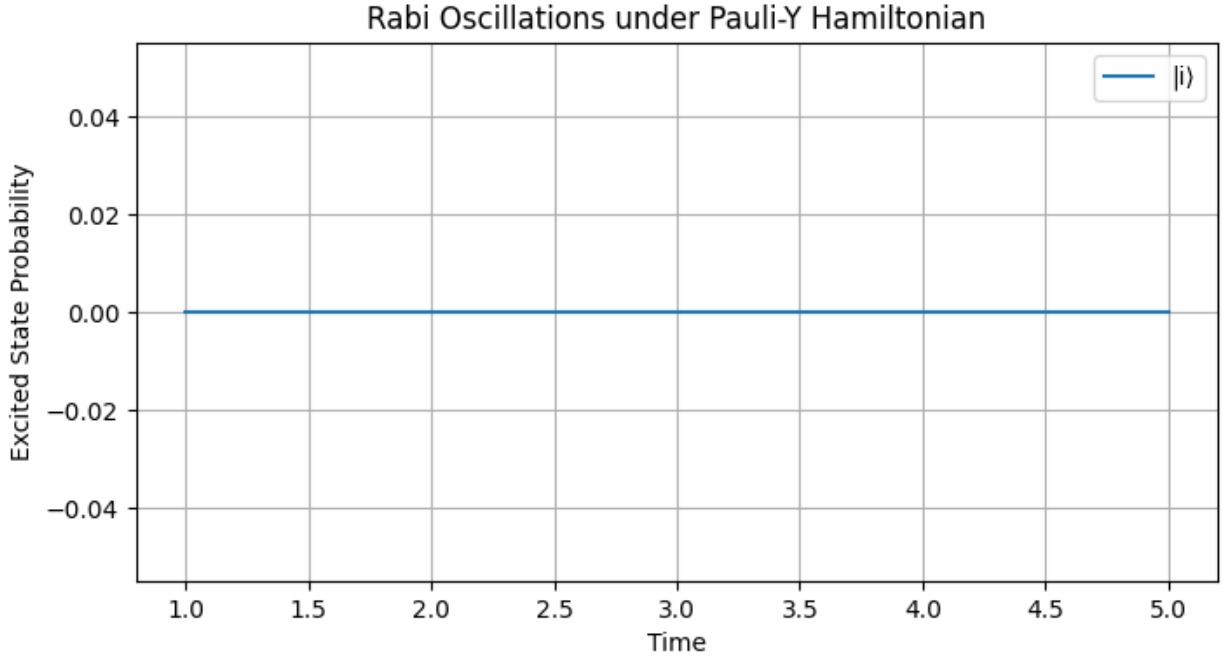

Rabi Oscillations under Pauli-X Hamiltonian
1e−12+5e−1
|-⟩
Excited State Probability
Time

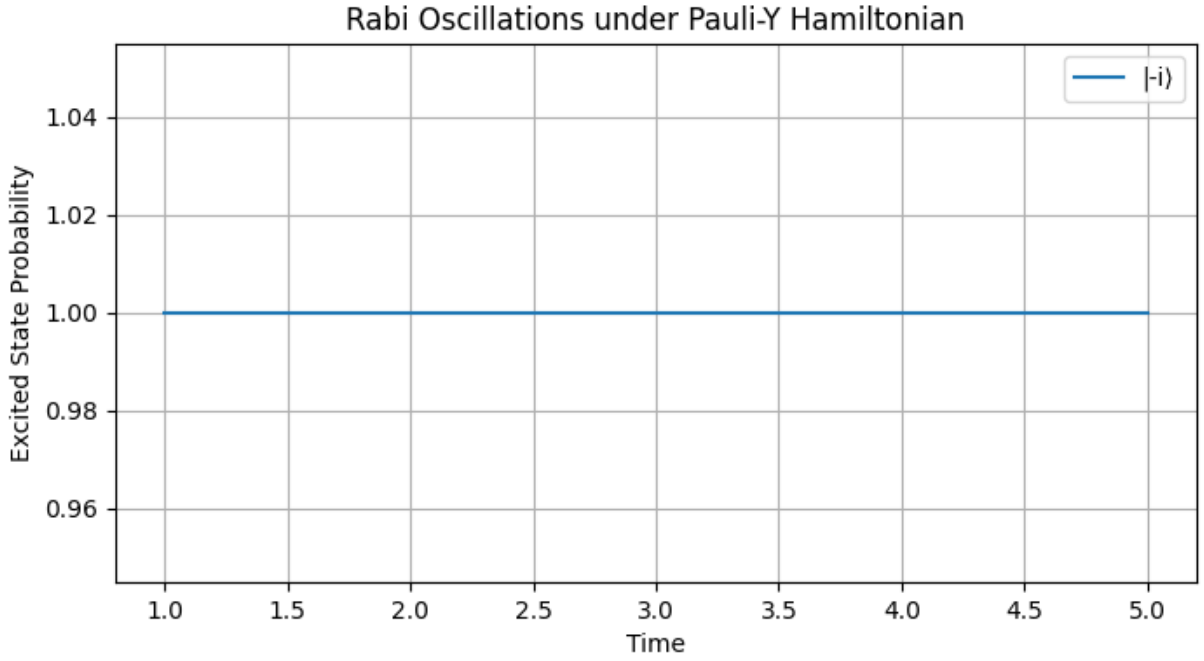

Rabi Oscillations under Pauli-Y Hamiltonian
|0⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

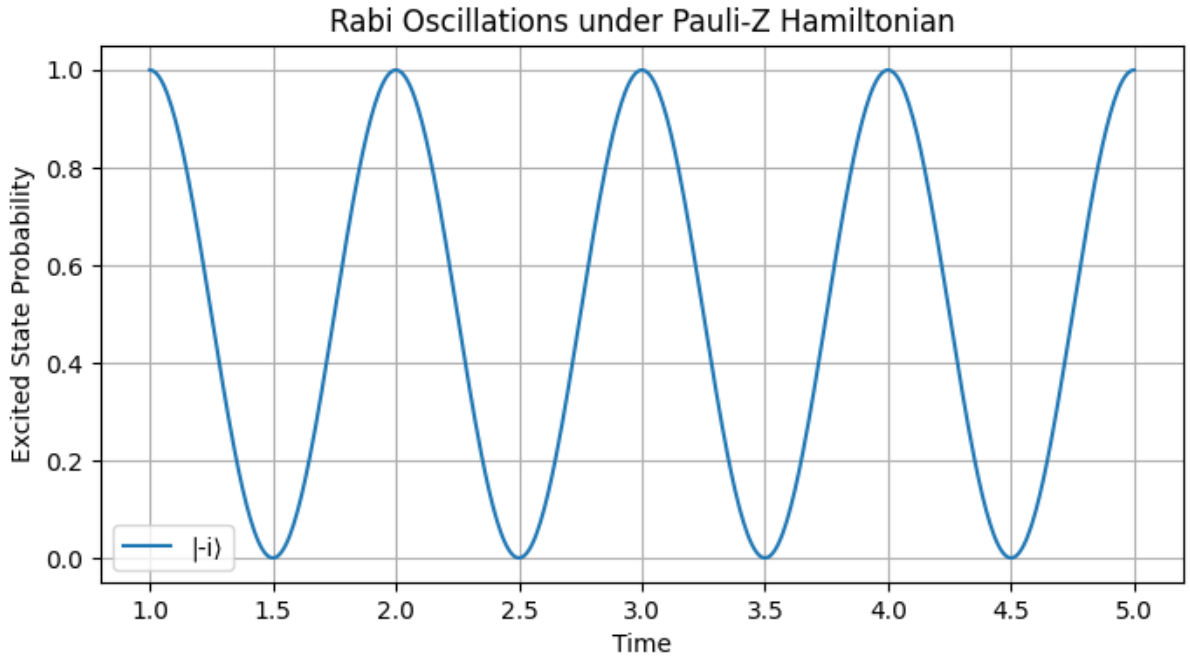

Rabi Oscillations under Pauli-Y Hamiltonian
|1⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

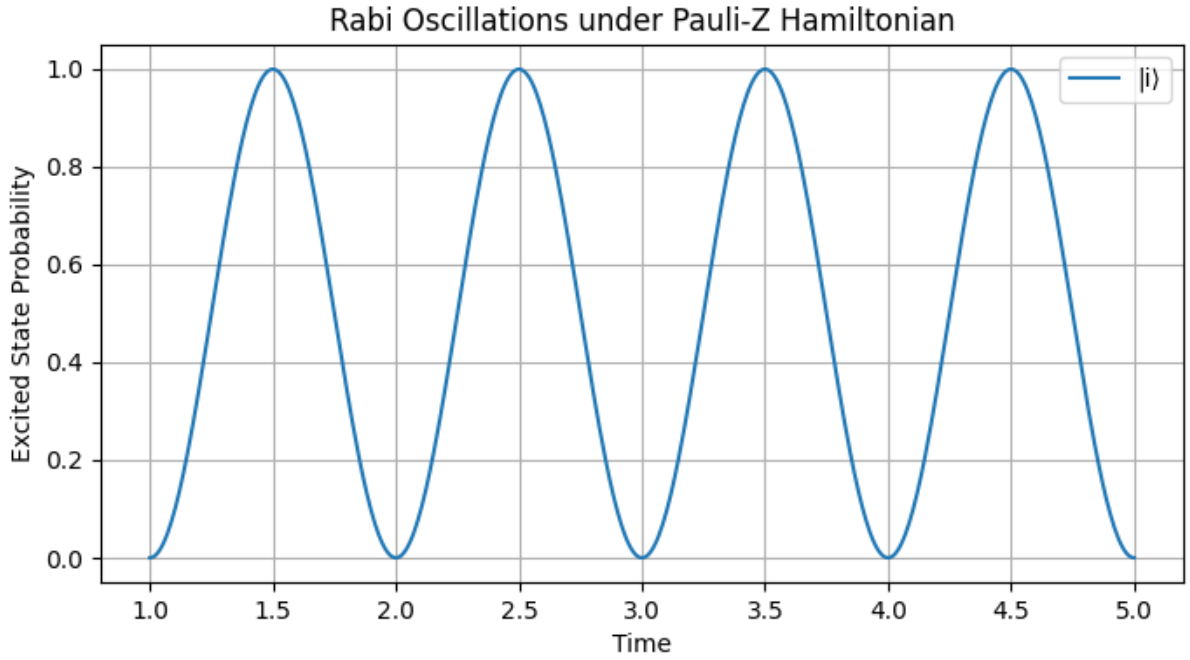

Rabi Oscillations under Pauli-Y Hamiltonian
|-⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

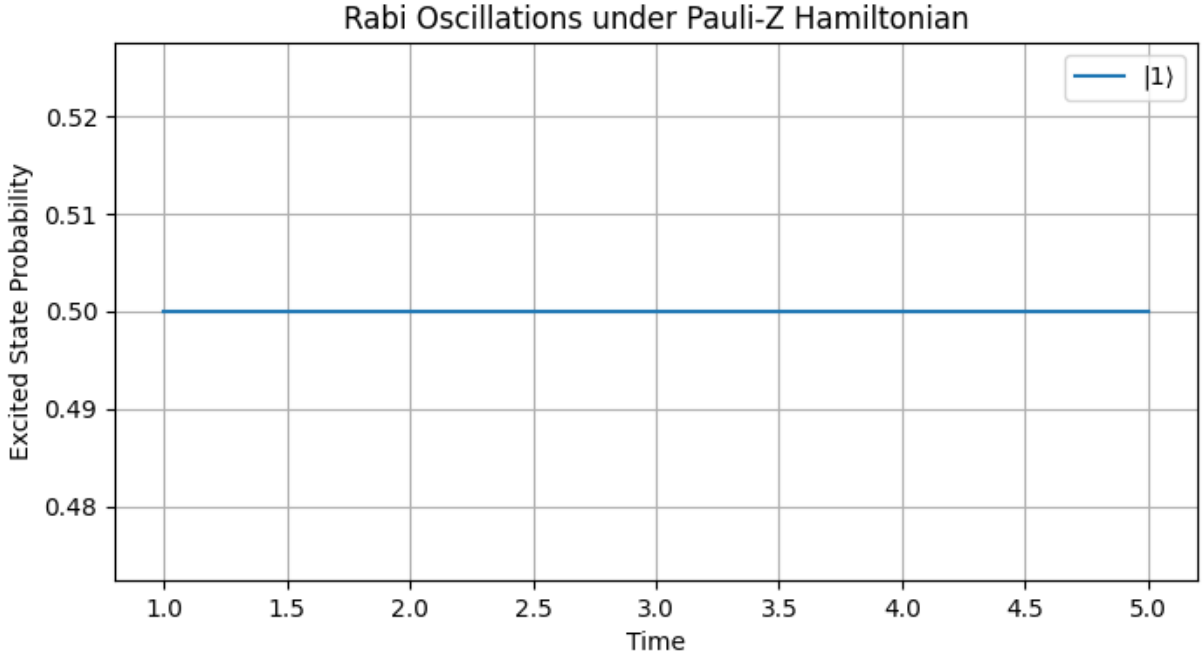

Rabi Oscillations under Pauli-Y Hamiltonian
|+⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

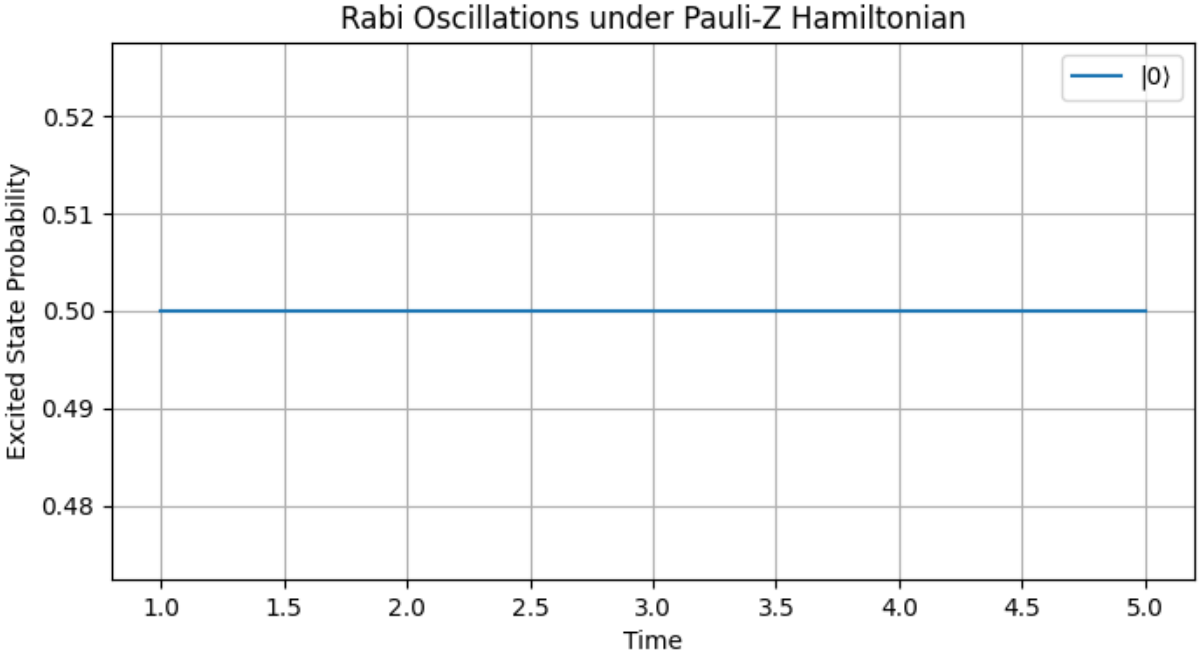

Rabi Oscillations under Pauli-Y Hamiltonian
|i⟩
Excited State Probability
0.04
0.02
0.00
−0.02
−0.04
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

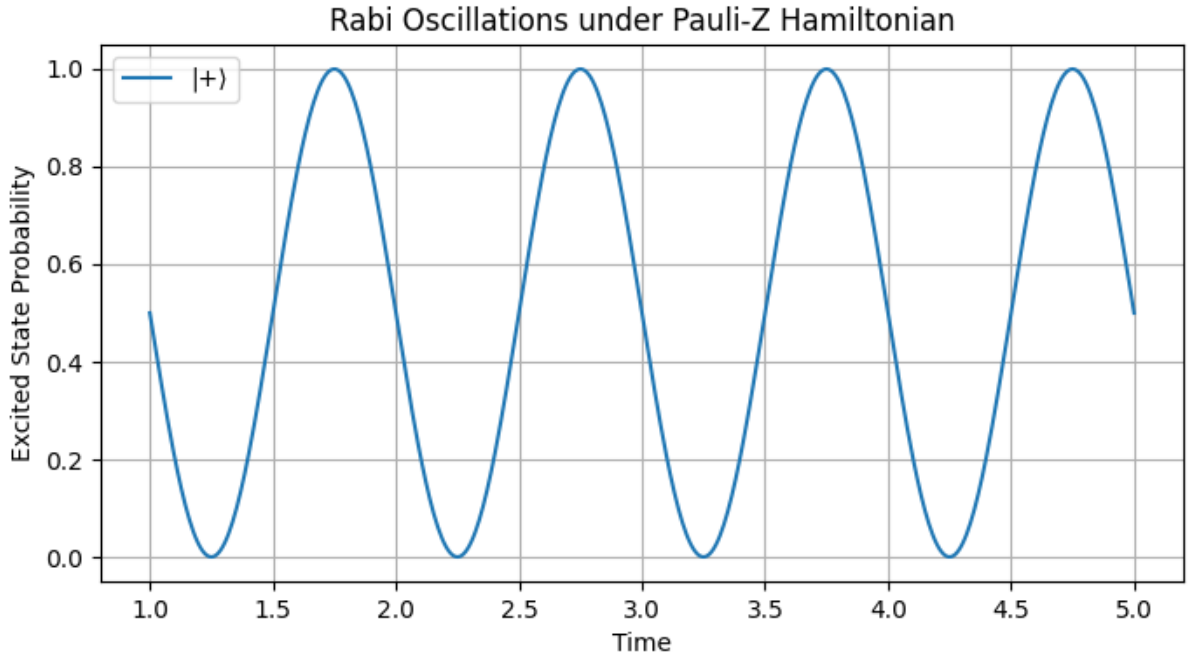

Rabi Oscillations under Pauli-Y Hamiltonian
|-i⟩
Excited State Probability
1.04
1.02
1.00
0.98
0.96
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

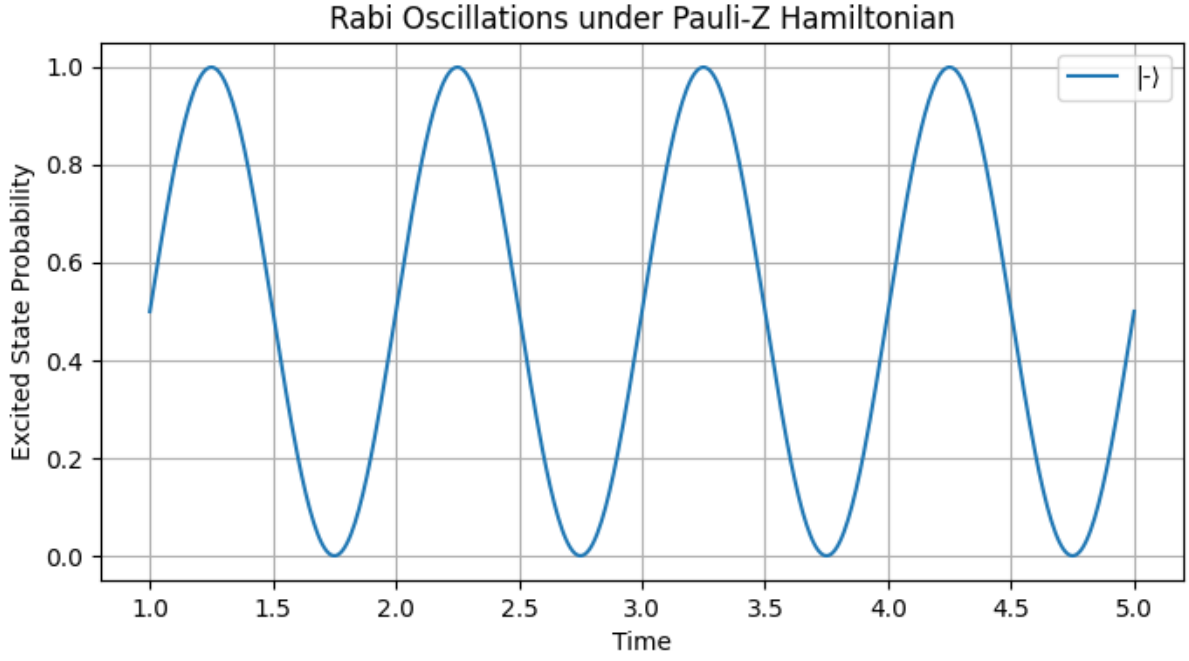

Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|-i⟩
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|i⟩
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|1⟩
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Rabi Oscillations under Pauli-Z Hamiltonian
|0⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

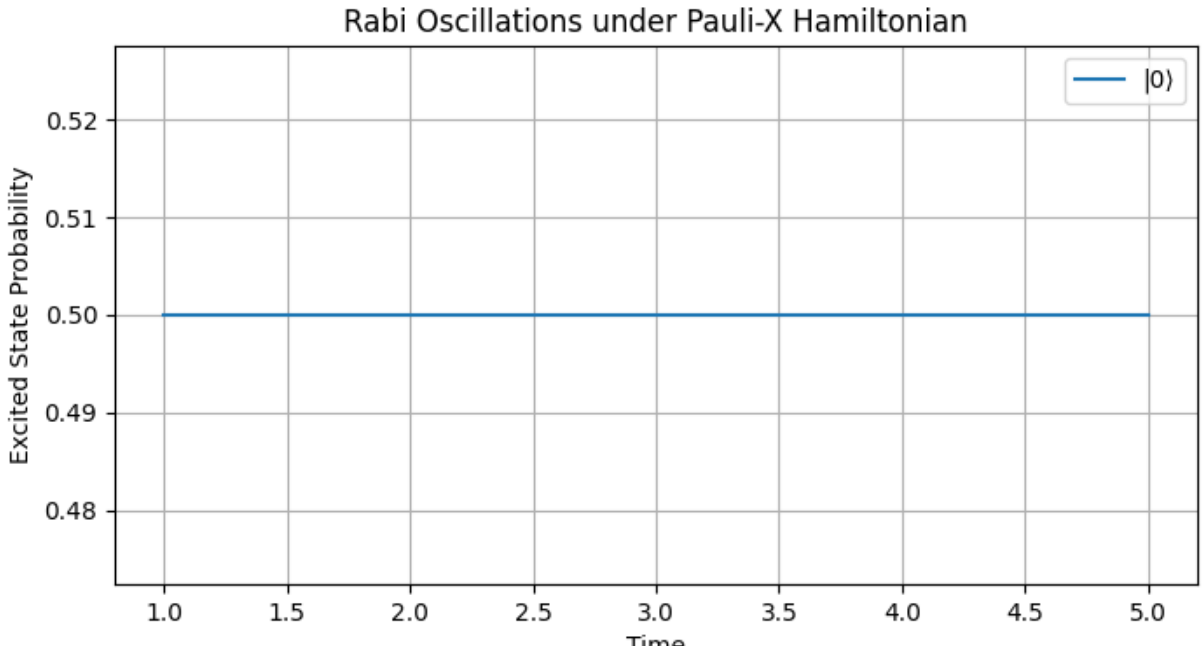

Rabi Oscillations under Pauli-Z Hamiltonian
|+⟩
Excited State Probability
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

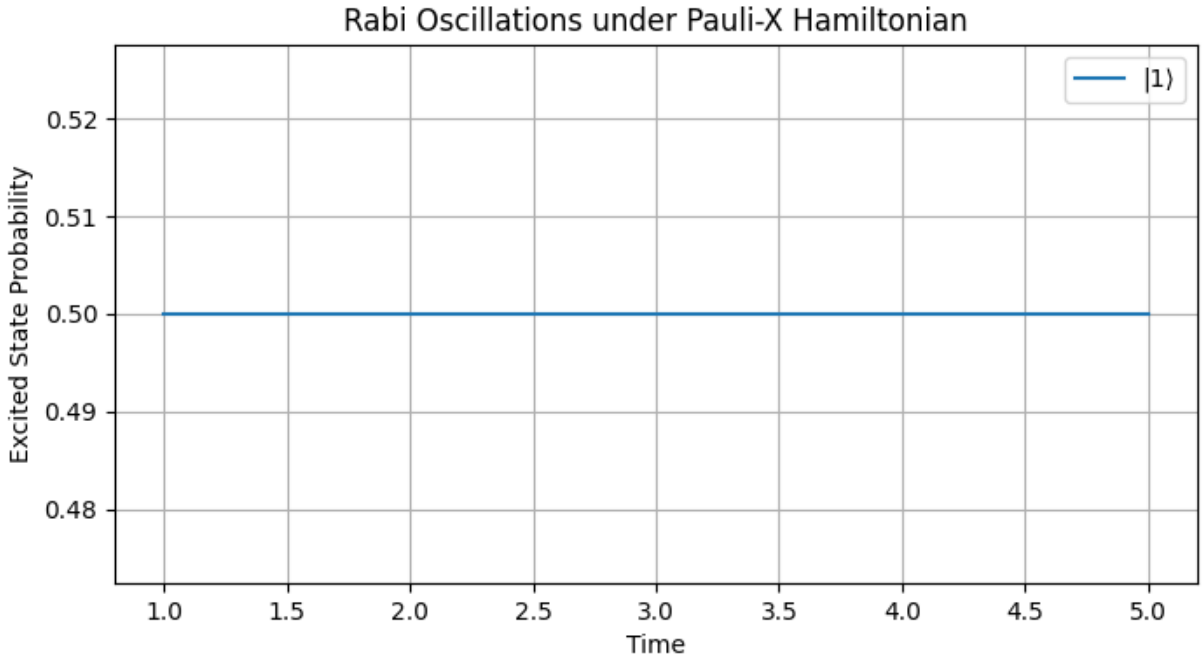

Rabi Oscillations under Pauli-Z Hamiltonian
|-⟩
Excited State Probability
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

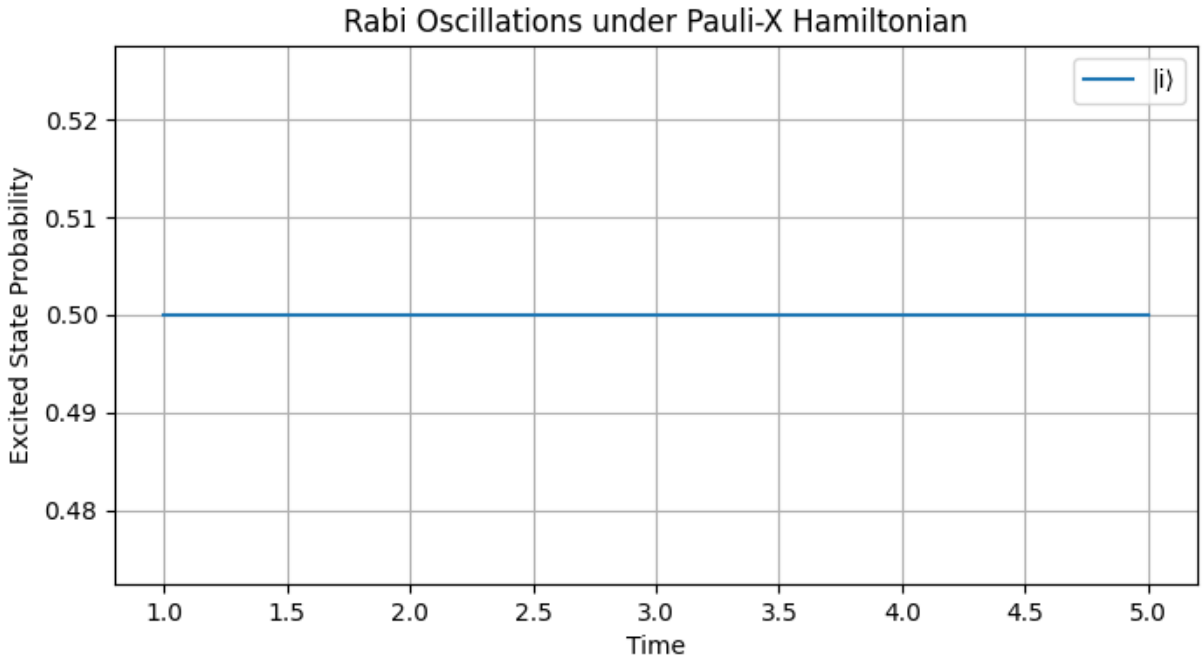
4.4.3 Probability of Excited State neg_state

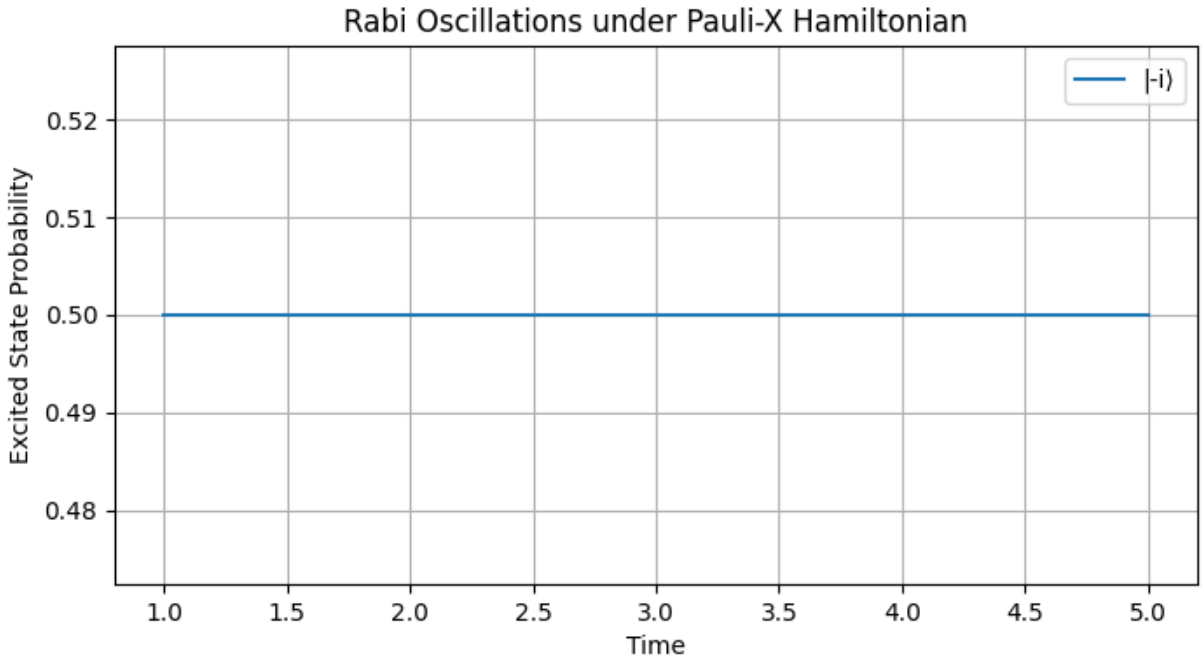
This program is exactly as outlined in 4.4.1, save Excited_Proj_init1 is swapped for neg_ state

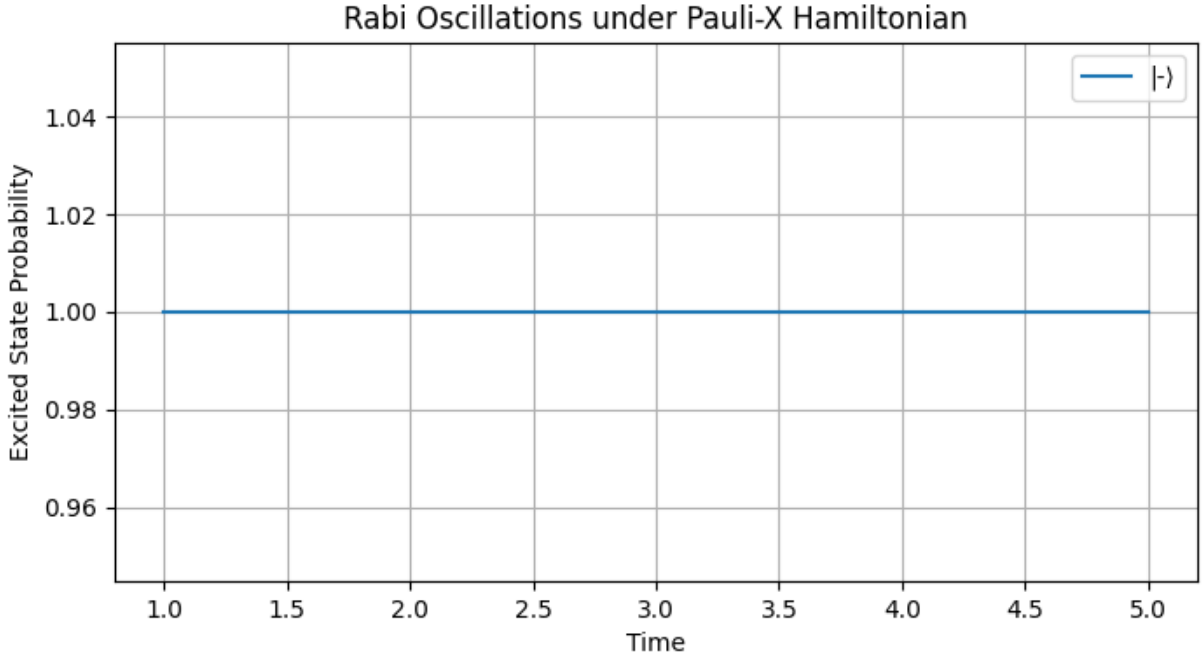
This program produces the following results

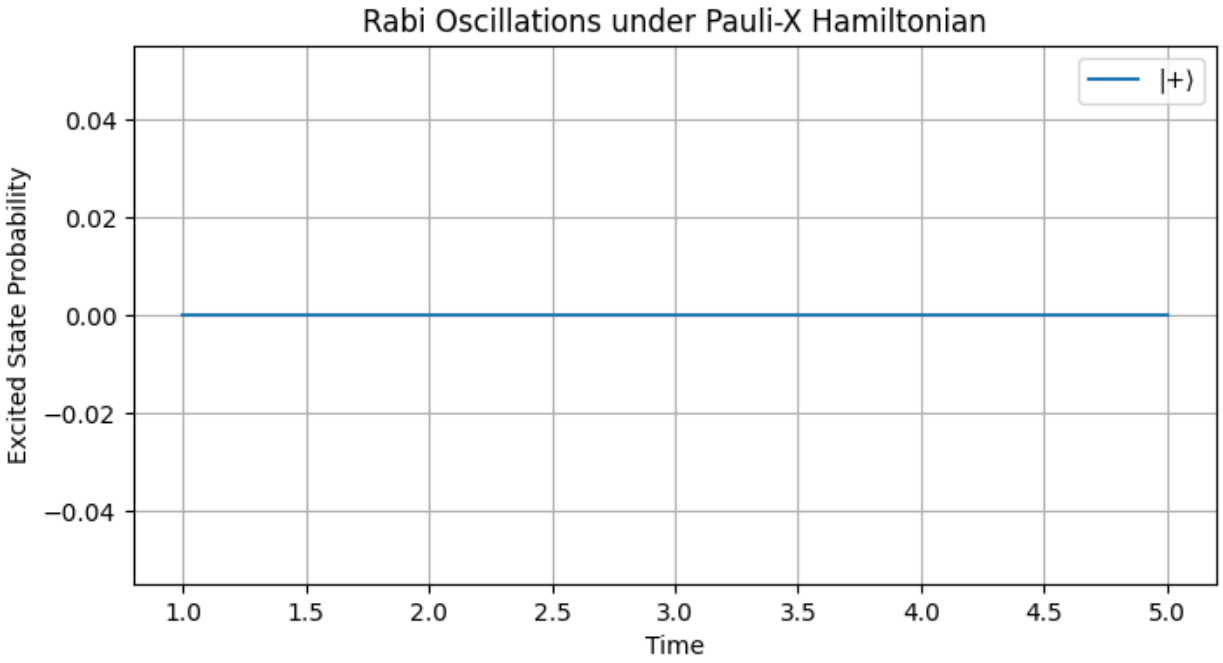


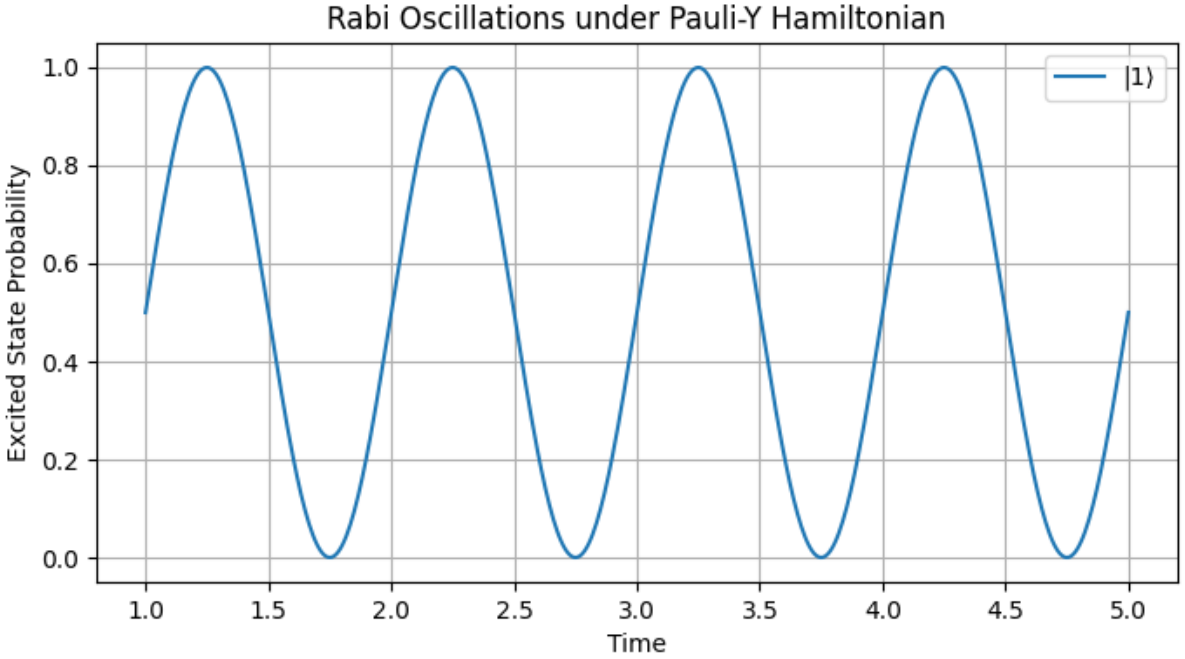


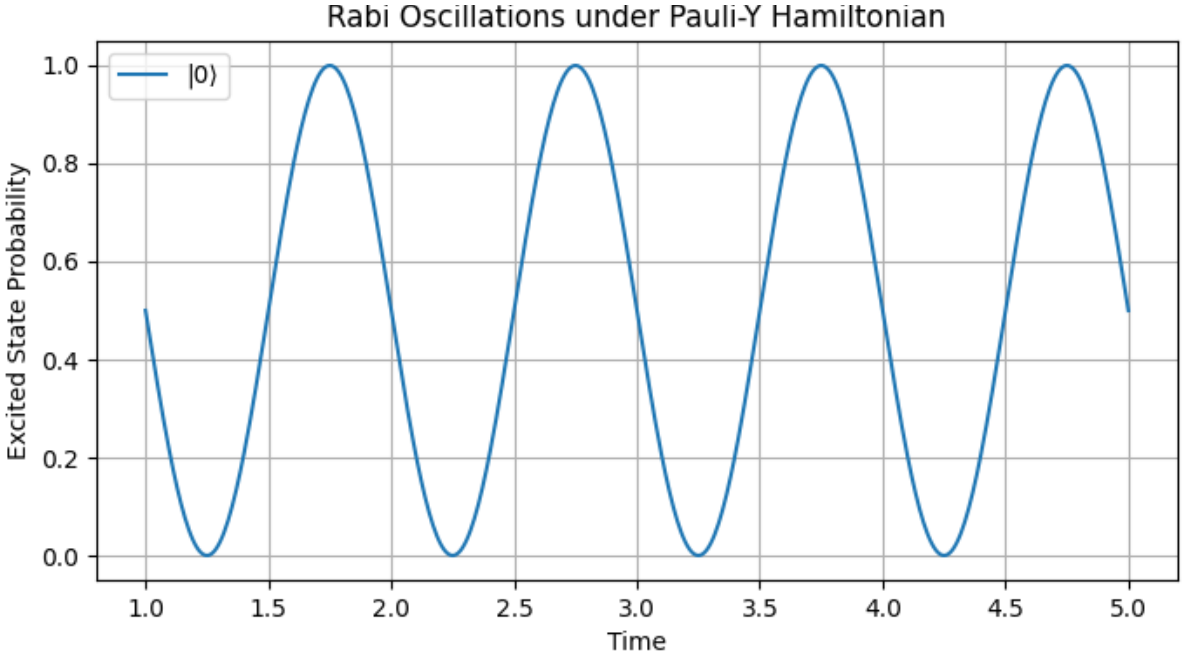

Rabi Oscillations under Pauli-X Hamiltonian
|i⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

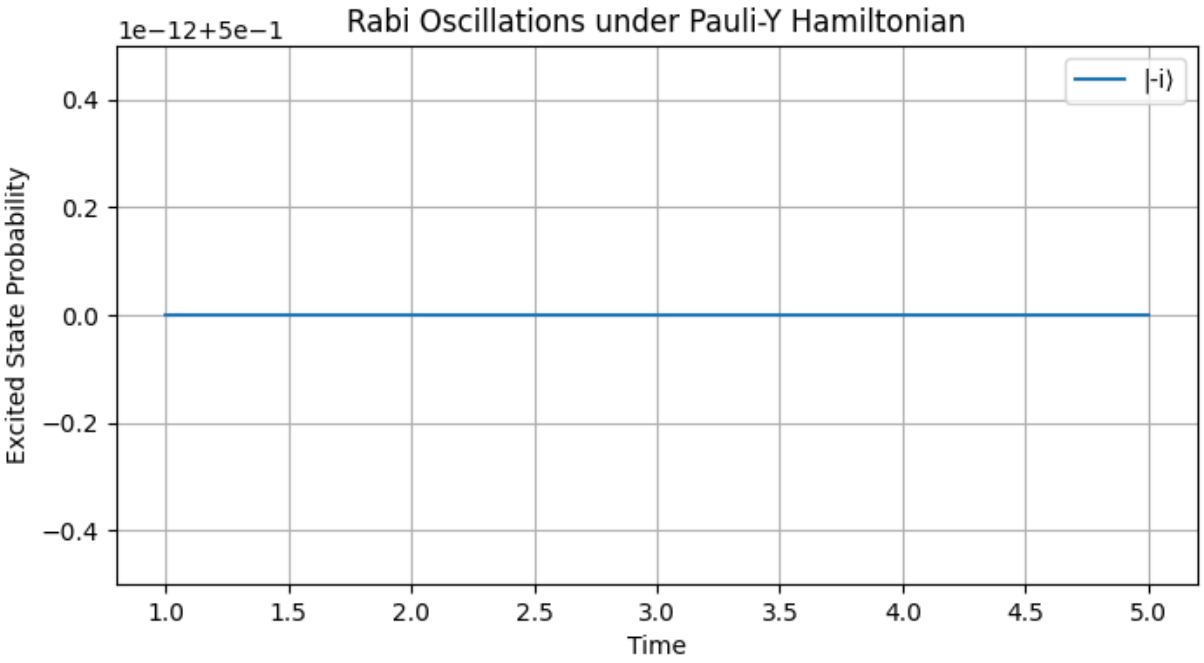

Rabi Oscillations under Pauli-X Hamiltonian
|-i⟩
Excited State Probability
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

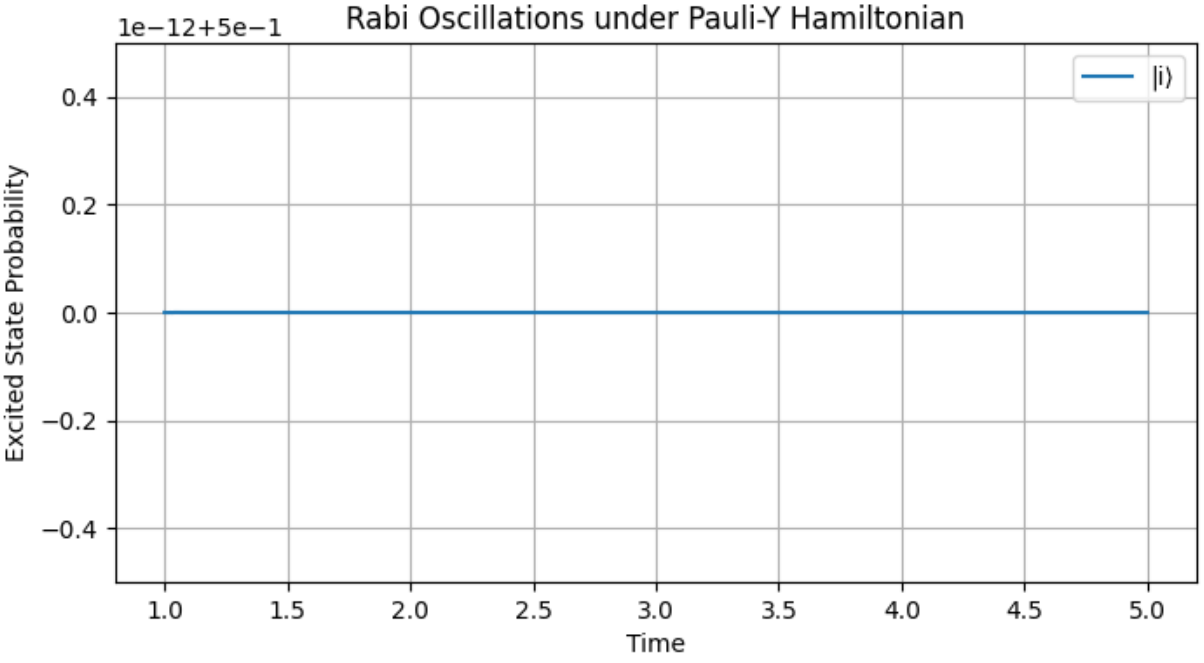

Rabi Oscillations under Pauli-X Hamiltonian
|-⟩
Excited State Probability
1.04
1.02
1.00
0.98
0.96
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

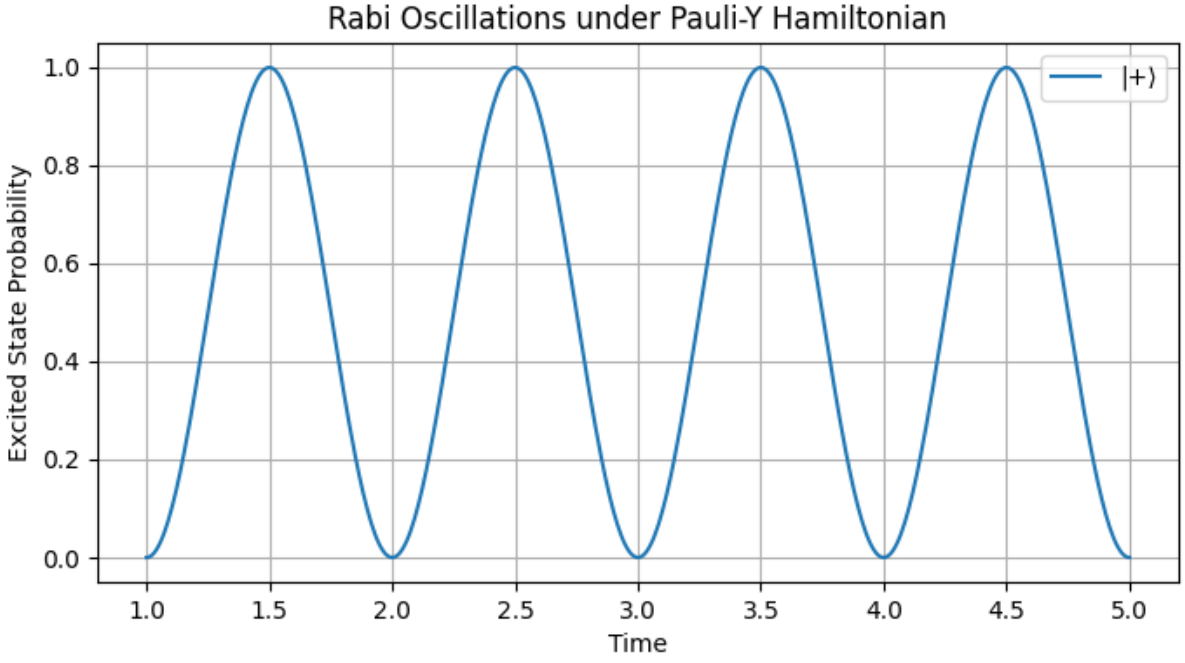

Rabi Oscillations under Pauli-X Hamiltonian
|+⟩
Excited State Probability
0.04
0.02
0.00
−0.02
−0.04
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

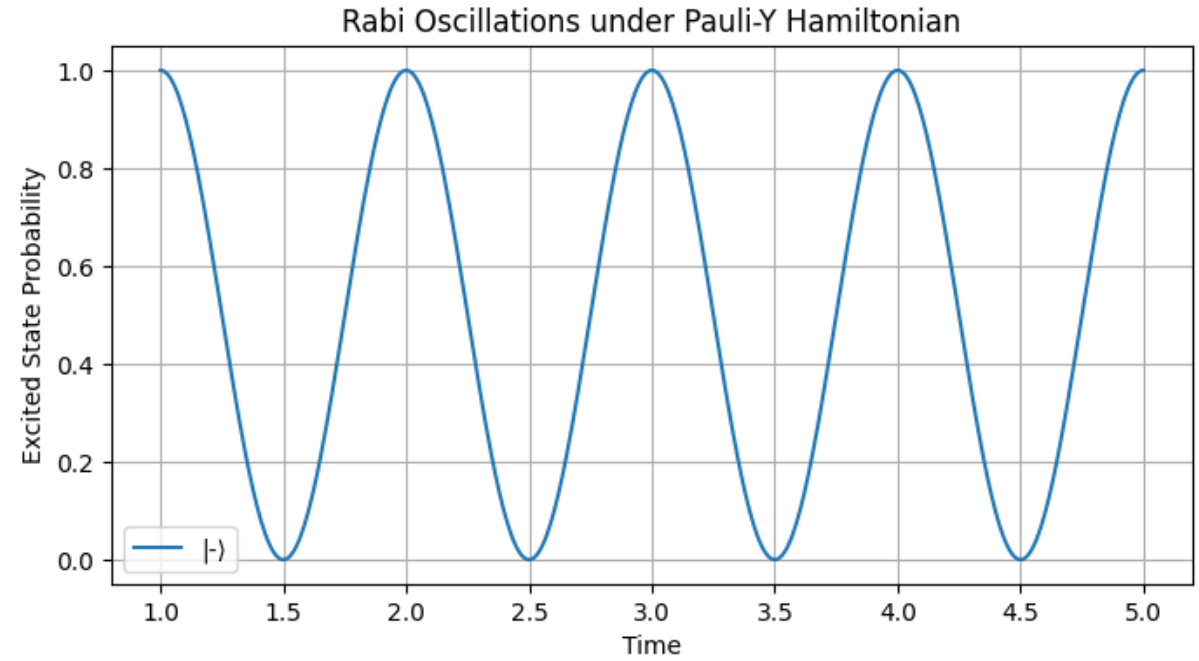

Rabi Oscillations under Pauli-Y Hamiltonian
|1⟩
Excited State Probability
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

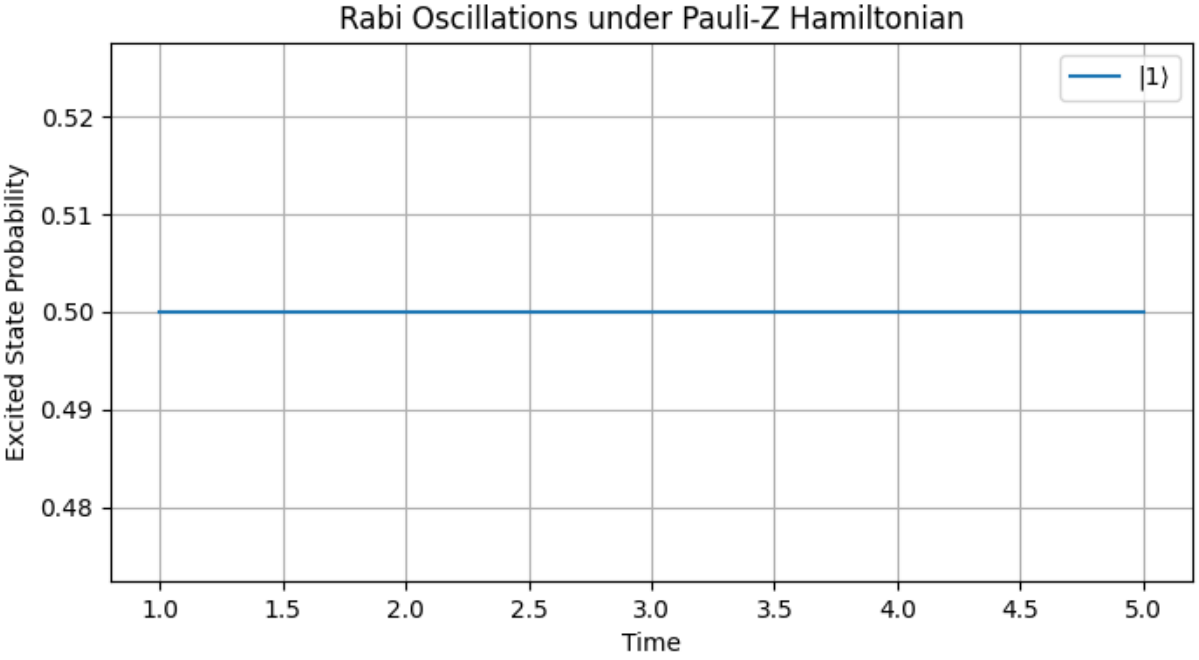

Rabi Oscillations under Pauli-Y Hamiltonian
|0⟩
Excited State Probability
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

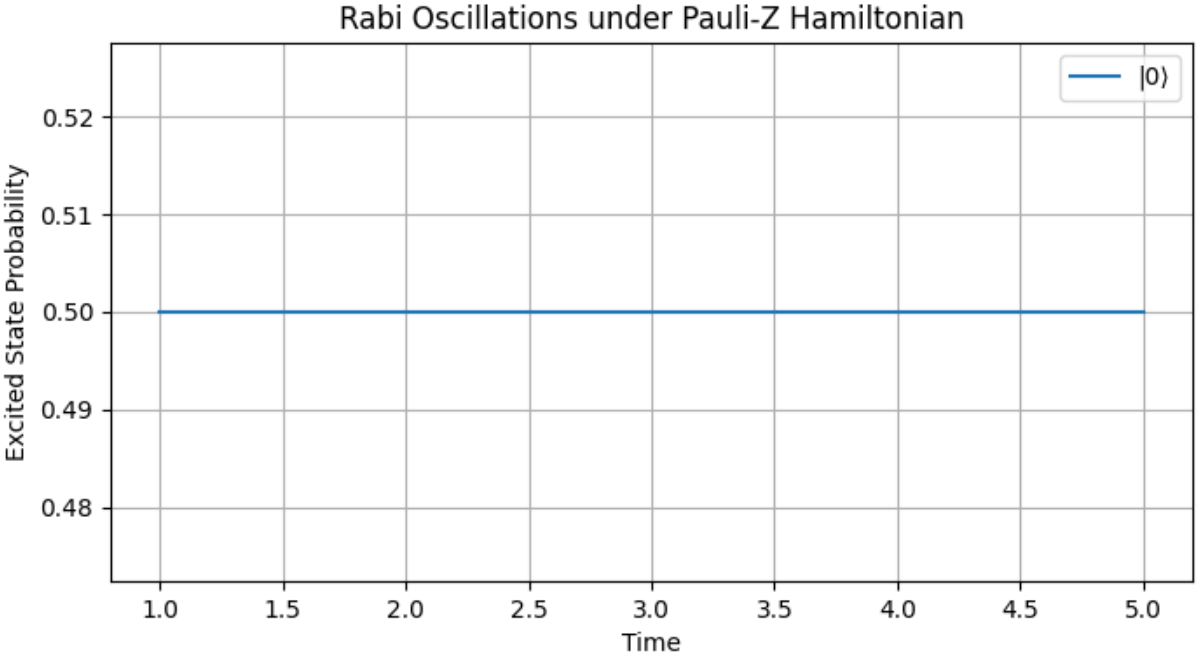

Rabi Oscillations under Pauli-Y Hamiltonian
1e−12+5e−1
|-i⟩
Excited State Probability
0.4
0.2
0.0
−0.2
−0.4
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

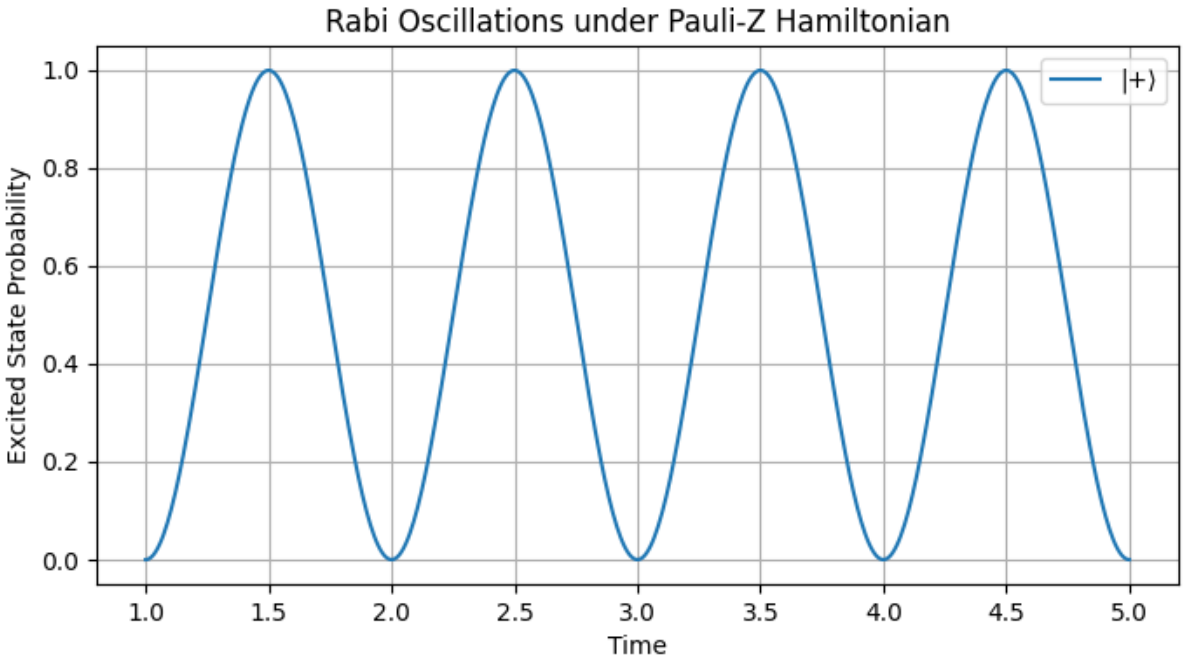

Rabi Oscillations under Pauli-Y Hamiltonian
1e−12+5e−1
|i⟩
Excited State Probability
0.4
0.2
0.0
−0.2
−0.4
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

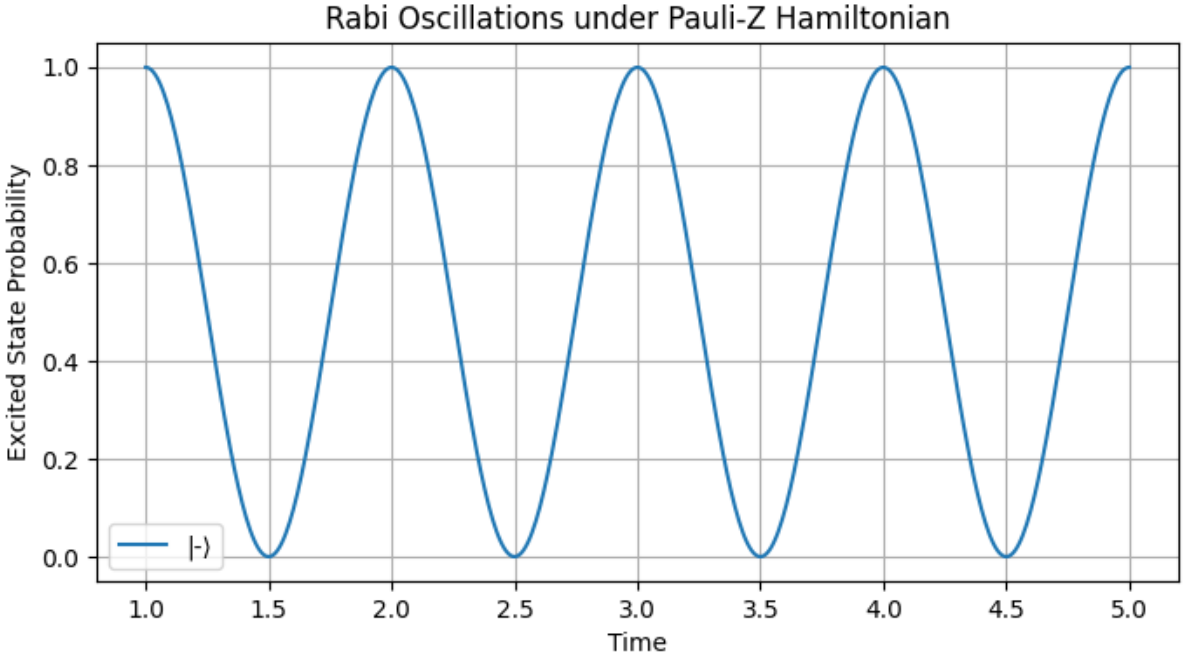

Rabi Oscillations under Pauli-Y Hamiltonian
|+⟩
Excited State Probability
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Time

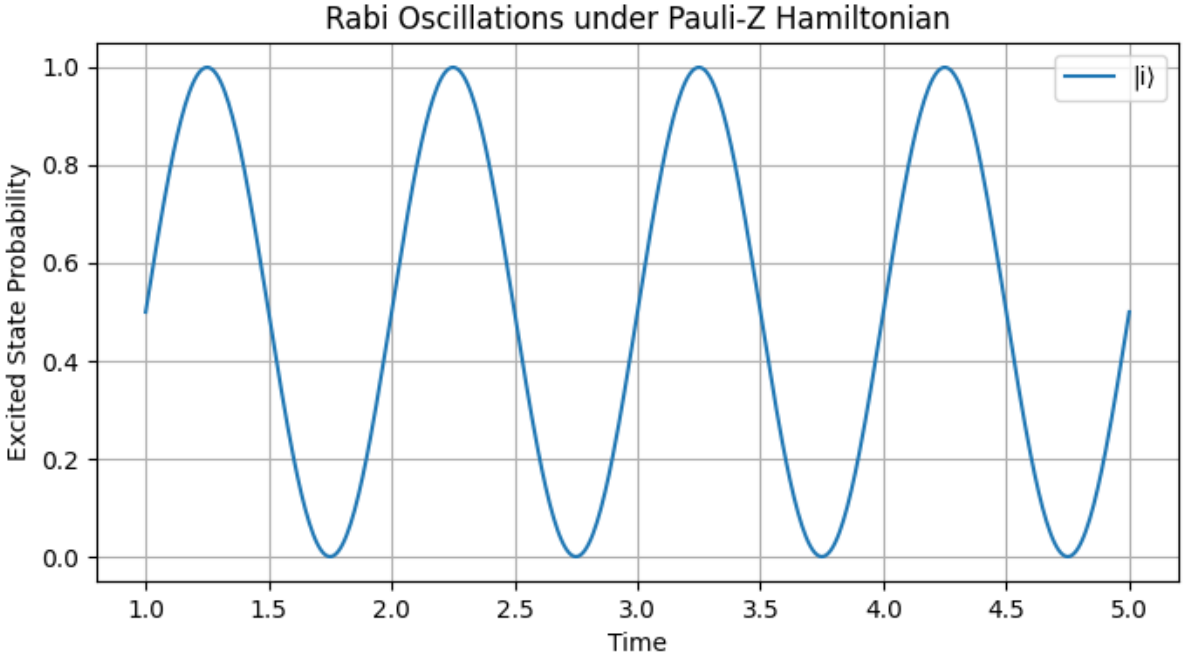

Rabi Oscillations under Pauli-Y Hamiltonian
Excited State Probability
Time
|-⟩
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

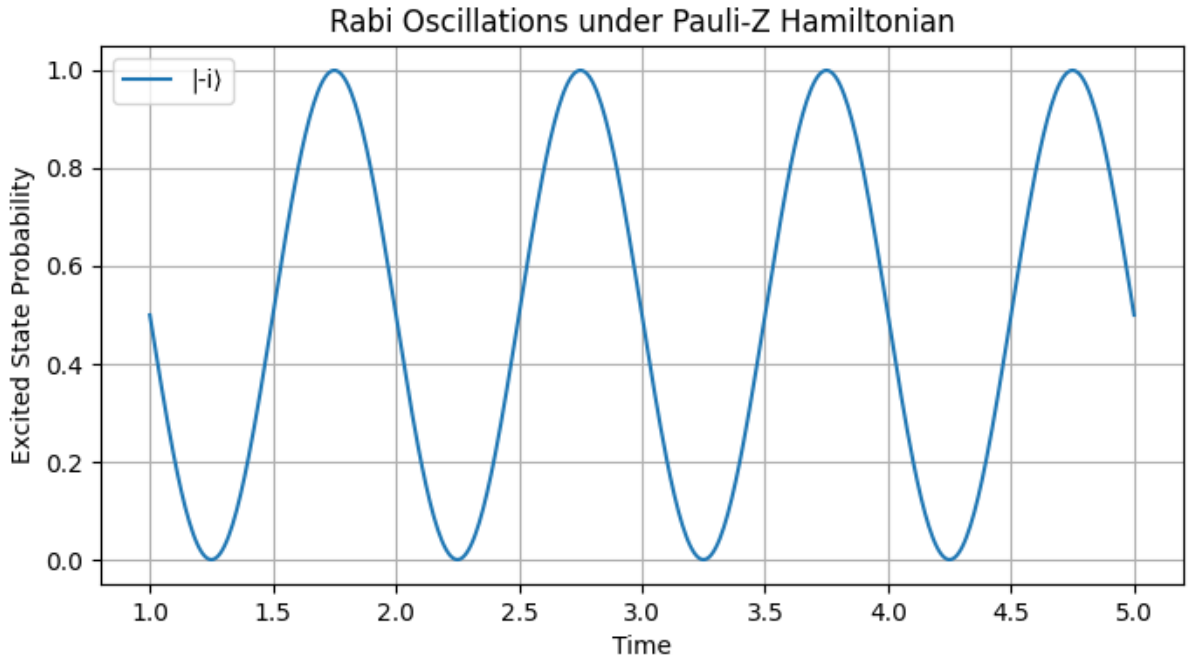

Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|1⟩
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|0⟩
0.52
0.51
0.50
0.49
0.48
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|+⟩
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|-⟩
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Rabi Oscillations under Pauli-Z Hamiltonian
Excited State Probability
Time
|i⟩
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



The reason I focused on measuring the probability of being in the states $|1\rangle$, $|-\rangle$, and $|-i\rangle$ is twofold:

1. These states are typically considered the excited states of the system, representing higher energy levels compared to their counterparts.
2. Measuring the complementary states $|0\rangle$, $|+\rangle$, and $|i\rangle$ would produce results that are simply mirror images of the chosen states and would not offer additional insight into the system's dynamics.

4.5 Analysing results

4.5.1 Bloch Spheres Analysis

Now we can compare the results to the expected outcomes outlined in 4.1.1.

On the Bloch sphere, a bit-flip is represented by the qubit tracing a circular path (or "ring") around the axis pointing toward the opposite basis state.

Quantum state/ Pauli Gates	Program 4.3.2	Program 4.3.3	Program 4.3.4
$\|0\rangle$	4.3.2_zero_state	4.3.3_zero_state	4.3.4_zero_state
$\|1\rangle$	4.3.2 – one_state	4.3.3 – one_state	4.3.4 – one_state
$\|+\rangle$	4.3.2 – plus_state	4.3.3 – plus_state	4.3.4 – plus_state
$\|-\rangle$	4.3.2 – neg_state	4.3.3 – neg_state	4.3.4 – neg_state
$\|i\rangle$	4.3.2 – i_state	4.3.3 – i_state	4.3.4 – i_state
$\|-i\rangle$	4.3.2 – neg_i_state	4.3.3 – neg_i_state	4.3.4 – neg_i_state

- Green represents where the results match the prediction
- Red represents where the results do not match the prediction

We see rings produced exactly where we predicted, providing sufficient evidence that the underlying principles have been understood and exercised.

4.5.2 2D graphs Analysis

For the 2D graphs, we generated all possible combinations of 6 initial states, 3 Hamiltonian operators, and 3 excited states, resulting in a total of 54 graphs. Each graph depicts the evolution of an initial state under the selected Hamiltonian and shows the probability of the system being measured in the specified excited state over time.

Below are the three tables produced by using all 2D graphs that show oscillation or no oscillations depending on the Ground state and Excited that have been set. The column are the ground states, and the row is the excited states

- Green = Oscillates between the Initial and excited state
- Red = No oscillation (flat line)

Pauli-Y	|1⟩	|-i⟩	|-⟩
|0⟩	4.4.1 - Pauli-Y init0 to excited init1	4.4.2 - Pauli-Y init0 to excited neg_i_state	4.4.3 - Pauli-Y init0 to excited neg_state
|1⟩	4.4.1 - Pauli-Y init1 to excited init1	4.4.2 - Pauli-Y init1 to excited neg_i_state	4.4.3 - Pauli-Y init1 to excited neg_state
|+⟩	4.4.1 - Pauli-Y plus_state to excited init1	4.4.2 - Pauli-Y plus_state to excited neg_i_state	4.4.3 - Pauli-Y plus_state to excited neg_state
|-⟩	4.4.1 - Pauli-Y neg_state to excited init1	4.4.2 - Pauli-Y neg_state to excited neg_i_state	4.4.3 - Pauli-Y neg_state to excited neg_state
|i⟩	4.4.1 - Pauli-Y i_state to excited init1	4.4.2 - Pauli-Y i_state to excited neg_i_state	4.4.3 - Pauli-Y i_state to excited neg_state
|-i⟩	4.4.1 - Pauli-Y i_neg_state to excited init1	4.4.2 - Pauli-Y i_neg_state to excited neg_i_state	4.4.3 - Pauli-Y neg_i_state to excited neg_state

Excited State

Pauli-X	|1⟩	|-i⟩	|-⟩
|0⟩	4.4.1 - Pauli-X init0 to excited init1	4.4.2 - Pauli-X init0 to excited neg_i_state	4.4.3 - Pauli-X init0 to excited neg_state
|1⟩	4.4.1 - Pauli-X init1 to excited init1	4.4.2 - Pauli-X init1 to excited neg_i_state	4.4.3 - Pauli-X init1 to excited neg_state
|+⟩	4.4.1 - Pauli-X plus_state to excited init1	4.4.2 - Pauli-X plus_state to excited neg_i_state	4.4.3 - Pauli-X plus_state to excited neg_state
|-⟩	4.4.1 - Pauli-X neg_state to excited init1	4.4.2 - Pauli-X neg_state to excited neg_i_state	4.4.3 - Pauli-X neg_state to excited neg_state
|i⟩	4.4.1 - Pauli-X i_state to excited init1	4.4.2 - Pauli-X i_state to excited neg_i_state	4.4.3 - Pauli-X i_state to excited neg_state
|-i⟩	4.4.1 - Pauli-X i_neg_state to excited init1	4.4.2 - Pauli-X i_neg_state to excited neg_i_state	4.4.3 - Pauli-X neg_i_state to excited neg_state

Pauli-Z	|1⟩	|-i⟩	|-⟩
|0⟩	4.4.1 - Pauli-Z init0 to excited init1	4.4.2 - Pauli-Z init0 to excited neg_i_state	4.4.3 - Pauli-Y init0 to excited neg_state
|1⟩	4.4.1 - Pauli-Z init1 to excited init1	4.4.2 - Pauli-Z init1 to excited neg_i_state	4.4.3 - Pauli-Y init1 to excited neg_state
|+⟩	4.4.1 - Pauli-Z plus_state to excited init1	4.4.2 - Pauli-Z plus_state to excited neg_i_state	4.4.3 - Pauli-Y plus_state to excited neg_state
|-⟩	4.4.1 - Pauli-Z neg_state to excited init1	4.4.2 - Pauli-Z neg_state to excited neg_i_state	4.4.3 - Pauli-Y neg_state to excited neg_state
|i⟩	4.4.1 - Pauli-Z i_state to excited init1	4.4.2 - Pauli-Z i_state to excited neg_i_state	4.4.3 - Pauli-Y i_state to excited neg_state
|-i⟩	4.4.1 - Pauli-Z i_neg_state to excited init1	4.4.2 - Pauli-Z i_neg_state to excited neg_i_state	4.4.3 - Pauli-Y neg_i_state to excited neg_state

Going by the data in the three tables above, it seems the expectations in 4.1.1 have been met.

- The Pauli-X Hamiltonian have driven Rabi oscillations between the Z and Y basis
- The Pauli-Y Hamiltonian have driven Rabi oscillations between the X band Z basis
- The Pauli-Z Hamiltonian have driven Rabi oscillations between the X and Y basis

Red Boxes

The red boxes mark graphs with a flat horizontal line, indicating that the selected Hamiltonian has no observable effect on the measured property—specifically, the probability of the qubit transitioning to the excited state. This is analogous to measuring only for the colour red: even if the system changes in other ways (e.g., height or turning blue), the reading remains zero because the measurement isn't sensitive to those changes.

Further analysis revealed a pattern: under the Pauli-X Hamiltonian, oscillations appear in both the Z and Y bases when either is used as the observable. This behaviour also holds for Pauli-Y and Pauli-Z Hamiltonians.

To show an example of how the red boxes are calculated, below is the calculation of the Pauli-Z Hamiltonian evolving the quantum state 0.

$$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}1\\ 0\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix} = |0\rangle \text{ , therefore } \sigma_z|0\rangle = |0\rangle$$

In some cases, a global phase does occur, like the Pauli-X basis and $|-\rangle$ state (which was predicted in 4.1)

$$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{bmatrix} = \begin{bmatrix}0 & -\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & 0\end{bmatrix} = \begin{bmatrix}-\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix} = -\begin{bmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{bmatrix} = -|-\rangle$$

But as detailed in 2.5.1.1, they make no physical changes to the qubit and therefore identical to the original state. Therefore, no evolution of the quantum state, producing a horizontal line on the graph.

Our predictions were confirmed by the resulting data and even led to additional insights. Specifically, we found that for the two bases in which we expected the Hamiltonian to cause evolution, oscillations can be observed in either basis, regardless of which one is used for measurement.

5 Conclusion and Final Analysis

This project set out to explore quantum simulation of physical systems by building a foundational understanding of quantum mechanics, implementing theoretical principles through code, and visually interpreting the behaviour of qubits under various Hamiltonians. Throughout the development process, both the theoretical framework and the computational implementation were closely aligned, allowing for meaningful simulations and observations.

Starting from first principles, the project carefully developed the mathematical language of quantum systems — from vector spaces and Hilbert spaces to inner and outer products — establishing a rigorous base for quantum computation. This was followed by translating abstract quantum concepts into computational models using Python and QuTiP, with the goal of simulating time evolution in two-level systems (qubits).

Simulations confirmed theoretical expectations, particularly the behaviour of quantum states under Pauli-X, Pauli-Y, and Pauli-Z Hamiltonians. The results demonstrated oscillatory behaviour of observables in the appropriate bases, validating predictions regarding Rabi oscillations and the effect of each Hamiltonian on different qubit states. Notably, consistent

patterns emerged showing that each Pauli-driven Hamiltonian produced measurable oscillations in the orthogonal bases, reinforcing the Bloch sphere interpretation of quantum dynamics.

Visually, the use of both 2D expectation value plots and 3D Bloch sphere renderings offered intuitive insights into the quantum state evolution. The project successfully modelled the dynamic rotation of qubits.

Despite being conducted in a classical environment, the simulation framework built here mimics genuine quantum behaviour, bridging theoretical concepts and practical computation. This project has not only deepened my understanding of the mathematical and physical structure of quantum mechanics but also provided a solid platform for exploring more advanced topics such as decoherence, entanglement, and multi-qubit systems in the future.

Ultimately, the project met its goals: to construct a simulation grounded in quantum theory, produce visual and measurable results, and validate key principles of qubit dynamics. It stands as a meaningful step forward in my journey toward mastering quantum computing.

Total word count (excluding references) : 11,957

6 References

1. Rieffel, E. and Wolfgang Polak (2011). Quantum computing: a gentle introduction. Cambridge, Mass.: Mit Press.
2. Chemistry LibreTexts. (2019). 7.13: The Three-Polarizer Paradox. [online] Available at: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Quantum_Tutorials_%28Rioux%29/07%3A_Quantum_Optics/7.13%3A_The_Three-Polarizer_Paradox?utm_source=chatgpt.com [Accessed 16 May 2025].
3. Axler, S. (1997). Linear Algebra Done Right. Springer Science & Business Media.