Quantum gates and quantum circuits and applications, summary from Monday

Nuclear TALENT course on quantum computing

Tuesday June 17, 2025

Quantum Control

The three basic steps:

1. Initialization (see for example https://journals.aps.org/prd/pdf/10.1103/PhysRevD.111.074515)
2. Evolution of the system
3. Measurement/readout

In this lecture we will illustrate this using some simple examples tailored to **quantum sensing**. We will also remind you of how we rephrase the physical processes in terms og gates and circuits. Let us start with a reminder from yesterday about various gates and how to encode them.

Widely used gates

There are several widely used quantum gates. Perhaps the most famous are the Pauli gates correspond to the Pauli matrices

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},$$

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

Algebra basis

These gates form a basis for the algebra $\mathfrak{su}(2)$. Exponentiating them will thus give us a basis for SU(2), the group within which all single-qubit gates live.

Exponentiated Pauli gates

These exponentiated Pauli gates are called rotation gates $R_\sigma(\theta)$ because they rotate the quantum state around the axis $\sigma = X, Y, Z$ of the Bloch sphere by an angle θ. They are defined as

$$R_X(\theta) = e^{-i\frac{\theta}{2}X} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix},$$

$$R_Y(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix},$$

$$R_Z(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}.$$

Basis for $\mathrm{SU}(2)$

Because they form a basis for $\mathrm{SU}(2)$, any single-qubit gate can be decomposed into three rotation gates. Indeed

$$R_z(\phi)R_y(\theta)R_z(\lambda) = \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \begin{bmatrix} e^{-i\lambda/2} & 0 \\ 0 & e^{i\lambda/2} \end{bmatrix}$$

which we can rewite as

$$e^{-i(\phi+\lambda)/2} \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\phi+\lambda)}\cos\frac{\theta}{2} \end{bmatrix},$$

which is, up to a global phase, equal to the expression for an arbitrary single-qubit gate.

Two-Qubit Gates

A two-qubit gate is a physical action that is applied to two qubits. It can be represented by a matrix U from the group SU(4). One important type of two-qubit gates are controlled gates, which work as follows: Suppose U is a single-qubit gate. A controlled-U gate (CU) acts on two qubits: a control qubit $|x\rangle$ and a target qubit $|y\rangle$. The controlled-U gate applies the identity I or the single-qubit gate U to the target qubit if the control gate is in the zero state $|0\rangle$ or the one state $|1\rangle$, respectively.

Control qubit

The control qubit is not acted upon. This can be represented as follows if

$$CU|xy\rangle = |xy\rangle \text{ if } |x\rangle = |0\rangle.$$

In matrix form

It is easier to see in a matrix form. It can be written in matrix form by writing it as a superposition of the two possible cases, each written as a simple tensor product

$$CU = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}.$$

CNOT gate

One of the most fundamental controlled gates is the CNOT gate. It is defined as the controlled-X gate CX. It can be written in matrix form as follows:

$$\mathrm{CNOT} = \mathrm{CX} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

CX gate

It changes, when operating on a two-qubit state where the first qubit is the control qubit and the second qubit is the target qubit, the states (check this)

$$\mathrm{CX}|00\rangle = |00\rangle,$$

$$\mathrm{CX}|10\rangle = |11\rangle,$$

$$\mathrm{CX}|01\rangle = |01\rangle,$$

$$\mathrm{CX}|11\rangle = |10\rangle,$$

which you can easily see by simply multiplying the above matrix with any of the above states.

Swap gate

A widely used two-qubit gate that goes beyond the simple controlled function is the SWAP gate. It swaps the states of the two qubits it acts upon

$$\mathrm{SWAP}|xy\rangle = |yx\rangle.$$

and has the following matrix form

$$\mathrm{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

An example of an OO code for quantum gates and circuits

```
import numpy as np
import random
import matplotlib.pyplot as plt
from collections import Counter
from mpl_toolkits.mplot3d import Axes3D # Ensure this is imported

# ================================= #
#       Quantum Gate Classes        #
# ================================= #

class Gate:
    def __init__(self, matrix, targets):
        self.matrix = np.array(matrix, dtype=np.complex128)
        # Convert Qubit objects (if any) to indices
        self.targets = [(t.index if isinstance(t, Qubit) else t) for t
        self.num_targets = len(self.targets)
        self.name = "CustomGate"

class OneQubitGate(Gate):
    def __init__(self, matrix, target):
        super().__init__(matrix, [target])
        # Add name attribute in subclasses if needed for representatio
        if np.array_equal(matrix, np.eye(2)): self.name = "I"
        elif np.array_equal(matrix, np.array([[0,1],[1,0]])): self.nam
        elif np.array_equal(matrix, np.array([[0,-1j],[1j,0]])): self.
        elif np.array_equal(matrix, np.array([[1,0],[0,-1]])): self.na
        elif np.allclose(matrix, (1/np.sqrt(2))*np.array([[1,1],[1,-1]
        elif np.array_equal(matrix, np.array([[1,0],[0,1j]])): self.na
```

Introduction to Qiskit

For the latest **qiskit** version see https://github.com/NuclearTalent/TalentQuantumComputingECT2025/blob/main/doc/pub/Exercises/1_getting_started_with_qiskit.ipynb

Basics of quantum sensing

In quantum sensing, **readout functions** refer to the methods used to extract information about the quantum system after a series of operations or measurements have been performed.

The readout process typically involves measuring the state of quantum bits (qubits) or other quantum observables, and this information is then used to infer physical quantities such as time, magnetic fields, temperature, or other parameters being sensed.

Example of Readout Functions in Quantum Sensing

A common example in quantum sensing involves measuring the state of a qubit after it has interacted with some external field or force. The quantum sensor's response is determined by how the qubit state evolves due to the applied field, and the readout function is the measurement procedure that allows us to extract this information.

Example 1: Quantum Magnetometry with a Single Qubit

Let's consider a typical scenario of quantum magnetometry using a single qubit that interacts with a magnetic field. We will use the concept of quantum state measurement as a readout function to infer the value of the magnetic field.

Step 1: Initial Qubit State

In this example, we use a spin-1/2 system (e.g., a qubit) that is sensitive to an external magnetic field. A common method of performing quantum sensing is to apply a quantum gate (e.g., a rotation) to the qubit and then measure its state.
Assume the initial state of the qubit is

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex amplitudes, and $|0\rangle$ and $|1\rangle$ are the computational basis states.

Step 2: Interaction with External Field

When a magnetic field is applied to the qubit, it will induce a phase shift in the qubit's state. The evolution of the qubit's state can be modeled using the time evolution operator $U(t)$, which depends on the interaction with the magnetic field. The evolution can be described as:

$$U(t) = \exp -\imath \mathcal{H} t$$

where $\mathcal{H}$ is the Hamiltonian of the qubit system, which in the case of a spin-1/2 particle in a magnetic field B could be

$$\mathcal{H} = -\gamma B \hat{S}_z,$$

with γ being the gyromagnetic ratio and $\hat{S}_z$ being the spin operator along the z-axis.

As time evolves

After time t, the qubit's state becomes:

$$|\psi(t)\rangle = \alpha|0\rangle + \beta \exp -(i\gamma Bt)|1\rangle,$$

Here, the phase shift $\exp -(i\gamma Bt)$ accumulates depending on the applied magnetic field B.

Step 3: Measurement (Readout Function)

After the interaction with the magnetic field, the next step is to measure the qubit. The readout function involves projecting the quantum state onto a measurement basis, often the computational basis $\{|0\rangle, |1\rangle\}$. The outcome of the measurement can then be used to infer information about the external field.

Measurement Probability

Measurement Probability: The probability of measuring the state $|0\rangle$ or $|1\rangle$ is given by the squared modulus of the corresponding amplitudes.

1. Probability of measuring $|0\rangle$: $P(0) = |\alpha|^2$
2. Probability of measuring $|1\rangle$: $P(1) = |\beta \exp -(i\gamma Bt)|^2 = |\beta|^2$

Readout Function

Readout Function: The measurement outcome corresponds to a result that is used to estimate the magnetic field. For instance, if the qubit undergoes a rotation due to the magnetic field, the phase shift in the qubit's state can be used to infer the magnetic field strength. Typically, a series of measurements and repeated experiments are performed to average out the noise and obtain a precise estimate of B.

Example 2: Using a Readout Function to Estimate Parameters

Let's say the qubit's state before the measurement is in a superposition state, and we want to determine the magnetic field strength by reading out the qubit's state. After the qubit has undergone the evolution $U(t)$, we measure it in the computational basis and repeat the experiment many times to obtain the average outcome.
We could perform parameter estimation by measuring the expected value of some observable (e.g., $\hat{S}_z$) and comparing the outcomes to theoretical predictions.

Expectation value of $\hat{S}_z$

For example, the expectation value of $\hat{S}_z$ (the spin along the z-axis) in the state $|\psi(t)\rangle$ is:

$$\langle \hat{S}_z \rangle = \langle \psi(t) | \hat{S}_z | \psi(t) \rangle = \frac{1}{2} \left(|\alpha|^2 - |\beta|^2 \right) ,$$

Phase shift

Since the magnetic field causes a phase shift $\exp -(i\gamma Bt)$, the measurement of the expectation value of $\hat{S}_z$ gives us a way to infer the magnetic field B. By performing a series of measurements and comparing the observed values to theoretical models, we can estimate the field strength B.

One-qubit system

In the first part of this example, we will analyze our systems using plain diagonalization and simple analytical manipulations. Thereafter we will develop codes and material for performing a quantum computing simulation of the same systems. Our first encounter is a simple one-qubit system, described by a simple 2×2 Hamiltonian. We start with a simple 2×2 Hamiltonian matrix expressed in terms of Pauli $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ matrices. But before we proceed, a simple reminder is appropriate.

Please not again: Single qubit gates

The Pauli matrices (and gate operations following therefrom) are defined as

$$\boldsymbol{X} \equiv \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{Y} \equiv \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \boldsymbol{Z} \equiv \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Pauli gates

The Pauli-$\boldsymbol{X}$ gate is also known as the **NOT** gate, which flips the state of the qubit.

$$\boldsymbol{X}|0\rangle = |1\rangle,$$
$$\boldsymbol{X}|1\rangle = |0\rangle.$$

The Pauli-$\boldsymbol{Y}$ gate flips the bit and multiplies the phase by i.

$$\boldsymbol{Y}|0\rangle = i|1\rangle,$$
$$\boldsymbol{Y}|1\rangle = -i|0\rangle.$$

The Pauli-$\boldsymbol{Z}$ gate multiplies only the phase of $|1\rangle$ by -1.

$$\boldsymbol{Z}|0\rangle = |0\rangle,$$
$$\boldsymbol{Z}|1\rangle = -|1\rangle.$$

Hadamard gate

The Hadamard gate is defined as

$$\boldsymbol{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

It creates a superposition of the $|0\rangle$ and $|1\rangle$ states.

$$\boldsymbol{H}|0\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle\right), \tag{1}$$

$$\boldsymbol{H}|1\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle - |1\rangle\right). \tag{2}$$

Note that we will use H as symbol for the Hadamard gate while we will reserve the notation $\mathcal{H}$ for a given Hamiltonian.

Sensing Hamiltonian

For our discussions, we will assume that the quantum sensor can be described by the generic Hamiltonian

$$\mathcal{H}(t) = \mathcal{H}_0 + \mathcal{H}_I(t) + \mathcal{H}_{\mathrm{control}}(t),$$

where $\mathcal{H}_0$ is the internal Hamiltonian, $\mathcal{H}_I(t)$ is the Hamiltonian associated with a signal ($V(t)$ in the notes below), and $\mathcal{H}_{\mathrm{control}}(t)$ is the control Hamiltonian. Following the above mentioned authors, we will assume that $\mathcal{H}_0$ is known and that $\mathcal{H}_{\mathrm{control}}(t)$ can be chosen so as to manipulate or tune the sensor in a controlled way. The goal of a quantum sensing experiment is then to infer $V(t)$ from the effect it has on the actual qubits via its Hamiltonian $\mathcal{H}_I(t)$, usually by a specific choice of $\mathcal{H}_{\mathrm{control}}(t)$.

Time-dependent Hamiltonian matrix

We define a hermitian matrix $\mathcal{H} \in \mathbb{C}^{2\times 2}$

$$\mathcal{H} = \begin{bmatrix} \mathcal{H}_{11} & \mathcal{H}_{12} \\ \mathcal{H}_{21} & \mathcal{H}_{22} \end{bmatrix}.$$

We let $\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_I$, where

$$\mathcal{H}_0 = \begin{bmatrix} E_0 & 0 \\ 0 & E_1 \end{bmatrix},$$

is a diagonal matrix. Similarly,

$$\mathcal{H}_I(t) = \begin{bmatrix} V_{11}(t) & V_{12}(t) \\ V_{21}(t) & V_{22}(t) \end{bmatrix},$$

where $V_{ij}(t)$ represent various time-dependent interaction matrix elements and since we have a hermitian matrix, we require that $V_{21} = V_{12}^*$.

Interaction part

We will now label the interaction matrix elements, assuming that they have an explicit time dependence. We define

$$\begin{aligned} V_{11} &= V_z(t) \\ V_{22} &= -V_z(t) \\ V_{12} &= V_x(t) - \imath V_y(t). \end{aligned}$$

In the numerical example below we let $V_y(t) = 0$, $V_z(t) = tV_z$ and $V_x(t) = tV_x$ with V_z and V_x being real-valued constants to be determined. In the same numerical example we let $t \in [0, 1]$.

Non-interacting solution

We can view $\mathcal{H}_0$ as the non-interacting solution

$$\mathcal{H}_0|0\rangle = E_0|0\rangle,$$

and

$$\mathcal{H}_0|1\rangle = E_1|1\rangle,$$

where we have defined the orthogonal computational one-qubit basis states $|0\rangle$ and $|1\rangle$.

Rewriting with Pauli matrices

We rewrite $\mathcal{H}$ (and $\mathcal{H}_0$ and $\mathcal{H}_I$) via Pauli matrices

$$\mathcal{H}_0 = \mathcal{E}_{\mathrm{avg}} \boldsymbol{I} - \Delta E \boldsymbol{Z}, \quad \mathcal{E}_{\mathrm{avg}} = \frac{E_0 + E_1}{2}, \; \Delta E = \frac{E_1 - E_0}{2},$$

and

$$\mathcal{H}_I = V_z(t)\boldsymbol{Z} + V_x(t)\boldsymbol{X} + V_y(t)\boldsymbol{Y},$$

with $V_z(t) = V_{11} = -V_{22}$, $V_x(t) = \Re(V_{12})$ and $V_y(t) = \Im(V_{12})$. This is the expression we will discuss in connection with quantum computing simulations. The discussions here, focus mainly on some simpler analytical considerations and simplifications. The numerical solutions are also given by standard eigenvalue solvers.

Simple time dependence

We let our Hamiltonian depend linearly on time t

$$\mathcal{H} = \mathcal{H}_0 + t\mathcal{H}_{\mathrm{I}},$$

with $t \in [0, 1]$, where the limits $t = 0$ and $t = 1$ represent the non-interacting (or unperturbed) and fully interacting system, respectively. This means that the various potential terms are given by $V_i(t) = tV_i$, with $i = \{x, y, z\}$ and V_i are real-valued constants.

Exact solution

Since this a simple 2×2 matrix eigenvalue problem we find the eigenvalues λ_0 and λ_1 to be

$$\lambda_{0,1} = \mathcal{E}_{\mathrm{avg}} \pm \Delta E \sqrt{1 + \frac{2V_z(t)}{\Delta E} + \frac{1}{\Delta E^2}(V_z^2(t) + V_x^2(t) + V_y^2(t))}.$$

If we assume that $\Delta E \gg V_z(t)$ and set $V_x(t) = V_y(t) = 0$ for simplicity and Taylor-expand our square root expression, we obtain

$$\lambda_0 = E_0 - \frac{1}{2}V_z(t),$$

$$\lambda_1 = E_1 + \frac{1}{2}V_z(t),$$

where we kept only terms up to ΔE. The above problem can however be easily solved numerically, see the code here.

Selecting parameters

The model is an eigenvalue problem with only two available states. Here we set the parameters $E_0 = 0$, $E_1 = 4$, $V_{11} = -V_{22} = 3$ and $V_{12} = V_{21} = 0.2$.
The non-interacting solutions represent our computational basis. Pertinent to our choice of parameters, is that at $t \geq 2/3$, the lowest eigenstate is dominated by $|1\rangle$ while the upper is $|0\rangle$. At $t = 1$ the $|0\rangle$ mixing of the lowest eigenvalue is 1% while for $t \leq 2/3$ we have a $|0\rangle$ component of more than 90%. The character of the eigenvectors has therefore been interchanged when passing $z = 2/3$. The value of the parameter V_{12} represents the strength of the coupling between the two states. Here we keep only the real part of the non-diagonal term.

Setting up the matrix for the simple one-qubit system

Here we solve the above problem as a standard eigenvalue problem (best seen using the jupyter-notebook)

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set_theme(font_scale=1.5)
from tqdm import tqdm

sigma_x = np.array([[0, 1], [1, 0]])
sigma_y = np.array([[0, -1j], [1j, 0]])
sigma_z = np.array([[1, 0], [0, -1]])
I = np.eye(2)

def Hamiltonian(lmb):
    E0 = 0
    E1 = 4
    V11 = 3
    V22 = -3
    V12 = 0.2
    V21 = 0.2

    eps = (E0 + E1) / 2
    omega = (E0 - E1) / 2
    c = (V11 + V22) / 2
    omega_z = (V11 - V22) / 2
    omega_x = V12

    H0 = eps * I + omega * sigma_z
```

Quantum control protocol, here tailored to sensing

1. The quantum sensor is initialized in some known basis, say $|0\rangle$.
2. The quantum sensor is transformed into the desired initial state $|\psi_{\text{Initial}}\rangle$, through an appropriate transformation $\boldsymbol{U}_1$. For a single qubit system this could be a Hadamard gate which results in a linear superposition of $|0\rangle$ and $|1\rangle$.
3. The quantum sensor evolves under the Hamiltonian $\mathcal{H}$ for a time t. At the end of the sensing period, the sensor is in its final stage (see below) $|\psi(t)\rangle = \boldsymbol{U}_{\mathcal{H}}(t, 0)\,|\psi_{\text{Initial}}\rangle$
4. This quantum state is transformed into a superposition of observable readout states, say a superposition of the one=qubit states $|0\rangle$ and $|1\rangle$, via the action $\boldsymbol{U}_2|\psi(t)$.
5. The final state of the quantum sensor is read out.
6. Steps 1-5 are repeated and averaged over a large number of cycles N to estimate the final transition probabilities p.
7. The transition probability p is measured as a function of time t and used to infer to desired signal $V(t)$.

Initialization and sensing analysis, Ramsey measurement

We stay now with the Taylor-approximated solution from the simple example above (the one-qubit case). We do so in order to illustrate some of the basic sensing ideas.
To initialize a given system to a known quantum state, we first start with a known initial state $|0\rangle$. Then, depending on the type of information that we want to learn about the stimulus, the measurement scheme to be used, and the physical implementation of the quantum system, we choose some unitary operator $\boldsymbol{U}_{\mathrm{Initial}}$ such that it transforms our state $|0\rangle$ to a desired initial superposition state $|\psi_{\mathrm{Initial}}\rangle = a|0\rangle + b|1\rangle$ for some $a, b \in \mathbb{C}$ such that $|a|^2 + |b|^2 = 1$.

Effects of stimulus

After the sensing state is initialized, it is exposed to the environment and evolves according to the time-evolution operator of the sensing Hamiltonian via the unitary transformation $\boldsymbol{U}_{\mathcal{H}}$ as (setting $\hbar = c = e = 1$)

$$|\psi(t)\rangle = \boldsymbol{U}_{\mathcal{H}}(t,0)\,|\psi_{\text{Initial}}\rangle\,.$$

In general we have

$$\boldsymbol{U}_{\mathcal{H}} = \exp\big(\imath\int_0^t \mathcal{H}(\tau)d\tau\big).$$

Here the Hamiltonian could be a complicated, non-analytical function with a time-dependent $V(t)$ (making $\mathcal{H}$ time-dependent as well).

Slowly changing potential

In the case where $V(t)$ is constant or changes much more slowly than our sensing integration time, we can assume

$$|\psi(t)\rangle = \boldsymbol{U}_{\mathcal{H}}(t,0)\,|\psi_{\mathrm{Initial}}\rangle = \exp\left(\imath t\mathcal{H}\right)|\psi_{\mathrm{Initial}}\rangle\,.$$

The sensing state evolves thus as

$$|\psi(t)\rangle = \left(\exp\imath t\left(E_0 - \frac{1}{2}V_z\right)|\lambda_0\rangle\langle\lambda_0| + \exp\imath t\left(E_1 + \frac{1}{2}V_z\right)|\lambda_1\rangle\langle\lambda_1|\right) \times |\psi_{\mathrm{Initial}}\rangle\,,$$

where we have using the spectral decomposition and the final representation of the sensing Hamiltonian found above.

Readout

After the sensing state has evolved over time in the presence of $V(t)$, it can be transformed again before a measurement is taken. The first part, the transformation to some desired read-out state, is performed by a readout operator, see discussions in Degen et al., 2017) where

$$|\psi_{\mathrm{Final}}\rangle = \boldsymbol{U}_{\mathrm{Readout}}|\psi(t)\rangle.$$

Here the readout operator $\boldsymbol{U}_{\mathrm{Readout}}$ is left unspecified.

Measurement

A measurement of this final state $|\psi_{\mathrm{Final}}\rangle = a'|0\rangle + b'|1\rangle$ is made with respect to the basis $\{|0\rangle, |1\rangle\}$ where $|0\rangle$ is measured with probability

$$|\langle 0 \mid \psi_{\mathrm{Final}}\rangle|^2 = \left|a'\right|^2,$$

and $|1\rangle$ is measured with probability

$$|\langle 1 \mid \psi_{\mathrm{Final}}\rangle|^2 = \left|b'\right|^2.$$

After this measurement, the sensing state has collapsed into one of the basis states and no more information can be gained.

Multiple measurements

However, by having multiple quantum sensing elements time-evolving together or by repeating the process many times before the external stimulus $V(t)$ can change, a transition probability

$$p_{|0\rangle\to|1\rangle} = \left|\langle 1 \mid \psi_{\mathrm{Final}}\rangle\right|^2 = \left|b'\right|^2,$$

can be estimated. The *sensing* is then achieved by taking a series of these transition probabilities as a time series, and then using the results to estimate the sensed stimulus $V(t)$

Example

The simplest mathematical example of quantum sensing is sensing an external stimulus's effect on the splitting of the energy levels of an isolated system. Suppose our stimulus is constant and *parallel* with our sensor, that is we set $V_z(t) = V_0$ and $V_x = 0$, and we choose our initialization and readout preparation operators to be the famous Hadamard gate

$$\boldsymbol{U}_H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Here, the subscript H stands for the Hadamard unitary transformation.

Evolution of initial state

The initial state is

$$|\psi_{\text{Initial}}\rangle = \boldsymbol{U}_H|0\rangle = \frac{1}{\sqrt{2}}\left[\begin{array}{cc} 1 & 1 \\ 1 & -1 \end{array}\right]\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}.$$

This may not necessarily be the same basis into which the system was initialized, but we assume it is so and then we only have to keep track of one basis.

State evolution

The state evolves as

$$
\begin{aligned}
|\psi(t)\rangle &= \left(\exp \imath t \left(E_0 - \frac{1}{2} V_z\right) |0\rangle\langle 0| + \exp \imath t \left(E_1 + \frac{1}{2} V_z\right) |1\rangle\langle 1|\right) |\psi_{\text{Initial}}\rangle \\
&= \frac{1}{\sqrt{2}} \exp \imath t \left(E_0 - \frac{1}{2} V_z\right) \begin{bmatrix} 1 \\ \exp \imath t \left(E_1 - E_0 + V_z\right) \end{bmatrix}
\end{aligned}
$$

Preparing for readout

This is then prepared for readout as

$$|\psi_{\mathrm{Final}}\rangle = \frac{1}{2} \exp \imath t (E_0 - \frac{1}{2} V_z) \begin{bmatrix} 1 + \exp \imath t (E_1 - E_0 + V_z) \\ 1 - \exp\left(\imath t (E_1 - E_0 + V_z)\right) \end{bmatrix}.$$

Transition probability

The transition probability

$$p_{|0\rangle\to|1\rangle} = |\langle 1 \mid \psi_{\mathrm{Final}}\rangle|^2 = |1 - \exp \imath t\,(E_1 - E_0 + V_z)|^2$$
$$= \frac{1}{2}\,(1 - \cos(t\,(E_1 - E_0) + V_z))$$

Ramsey interferometry

We know the difference in energy between E_1 and E_0, either since we constructed the system or by taking measurements without the external stimulus V, and we can control the time t for which the system is allowed to evolve under the external stimulus. Then we can fix t and take many measurements to estimate $p_{|0\rangle \to |1\rangle}$, which then makes finding tV_z a simple phase-estimation problem which gives us V_z. The physical implementation of this process is known as Ramsey Interferometry, and it can be done with arbitary initialization and readout preparation unitary operators.

Benefits of Entanglement

Up until now, we have said that we take many measurements of $|\psi_{\text{Final}}\rangle$ to estimate $p_{|0\rangle\rightarrow|1\rangle}$, but we have neglected the estimation process. Assuming we can take N measurements, either by having N experimental apparatuses running in parallel or by taking N different measurements of a (relatively) constant V with a single apparatus (this is what we will do below), the uncertainty in p, denoted as σ_p (this is a positive real number; not to be confused with the Pauli matrices), scales as

$$\sigma_p \propto \frac{1}{\sqrt{N}}$$

Ramsey interferometry

If we consider Ramsey Interferometry as an example, see `https://en.wikipedia.org/wiki/Ramsey_interferometry`, then the uncertainty in V_z, denoted σ_V, scales as

$$\sigma_V \propto \sigma_p \propto \frac{1}{\sqrt{N}}$$

This relationship is known as the standard quantum limit (SQL) , see Giovannetti *et al.*, 2011, `https://www.nature.com/articles/nphoton.2011.35`, but can also be explained with the law of Large Numbers from statistics, where measuring N similarly distributed, well-behaved random variables gives the sample mean as an estimator for the population mean and the sample variance divided by the size of the sample as an uncertainty in the estimate of the population mean.

Key components of code for one qubit

Physical System:

1. Hamiltonian: $\mathcal{H} = -\frac{\gamma B}{2} Z$ (we will set $\gamma = 1$)
2. Initial state: $|\psi_{\text{Initial}}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, Hadamard gate

Time Evolution:

1. Calculated using matrix exponential: $U_{\mathcal{H}}(t) = \exp -i\mathcal{H}t$
2. State at time t: $|\psi(t)\rangle = U_{\mathcal{H}}(t)|\psi_{\text{Initial}}\rangle$

Expectation Values:

1. $\langle X \rangle$: Oscillates as $\cos(Bt)$
2. $\langle Y \rangle$: Oscillates as $-\sin(Bt)$
3. $\langle Z \rangle$: Remains constant at 0

One-qubit program example (best seen using the jupyter-notebook)

```python
import numpy as np
from scipy.linalg import expm
import matplotlib.pyplot as plt

# Define single-qubit operations, Identity, Pauli, Hadamard and S matr
Id = np.array([[1, 0], [0, 1]], dtype=complex)
X = np.array([[0, 1], [1, 0]], dtype=complex)
Y = np.array([[0, -1j], [1j, 0]], dtype=complex)
Z = np.array([[1, 0], [0, -1]], dtype=complex)
Had = np.array([[1, 1],[1, -1]], dtype=complex) / np.sqrt(2)
S = np.array([[1, 0],[0, 1j]], dtype=complex)

# Simulation parameters
B = 2 * np.pi  # Magnetic field strength (in angular frequency units)
times = np.linspace(0, 1, 200)  # Time range from 0 to 1 seconds

# Initial state: superposition state via Hadamard gate acting on |0>
psi0 =  np.array([1, 0], dtype=complex) # start with |0> then act with
psi0 = Had @ psi0

# Lists to store expectation values
expect_x, expect_y, expect_z = [], [], []

for t in times:
    # Construct Hamiltonian
    H = B * Z/ 2
```

Interpretation

1. The oscillations in X and Y components demonstrate the Larmor precession caused by the magnetic field
2. The frequency of oscillation is directly proportional to the magnetic field strength B
3. This forms the basis for quantum sensing: measuring oscillation frequency allows precise determination of B

To use this for sensing (protocol of Degen *et al.*, (2017))

1. Prepare the qubit in a known superposition state
2. Let it evolve in the magnetic field for a known time
3. Measure the expectation values
4. Determine (here) B from the oscillation frequency

Feel free to play around with this code

1. Change *B* value to see different oscillation frequencies
2. Adjust *time* array to observe different numbers of oscillations
3. Add noise to simulate real-world sensing scenarios
4. Implement actual measurement simulations instead of expectation values

This code provides a fundamental demonstration of quantum sensing principles using a simple qubit system. Real-world implementations would typically use more sophisticated techniques like Ramsey interferometry or dynamical decoupling for enhanced sensitivity.

More than one qubit

The nature of quantum systems allows for more information to be extracted by exploiting entanglement between quantum systems. This is the fundamental basis for the benefits of quantum computing over classical computing, and quantum sensing has similar benefits over classical sensing. Suppose we return to the example above, but rather than initializing N sensing qubits separately, we initialize $\frac{N}{n}$ groups each with n entangled quantum systems. Then we have

$$|\psi_{\text{Init}}\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle^{\otimes n} + |1\rangle^{\otimes n}\right),$$

where $|0\rangle^{\otimes n} = |0\rangle \otimes \ldots \otimes |0\rangle$, n times.

After initialization

After initialization, each of the n sensing qubits evolves to pick up a relative phase factor of $\exp \imath t \left(E_1 - E_0 + V_z\right)$, which combined results in

$$|\psi(t)\rangle = \mathcal{N} \left(|0\rangle^{\otimes n} + \exp n\imath t \left(E_1 - E_0 + V_z\right)|1\rangle^{\otimes n}\right)$$

where $\mathcal{N}$ is just a factor to take care of normalization.

Transition probability

The transition probability

$$p_{|0\rangle \to |1\rangle} = |\langle 1 \mid \psi_{\text{Final}} \rangle|^2 = \frac{1}{2}\left(1 - \cos\left(tn\left(E_1 - E_0\right) + nV_z\right)\right)$$

Role of entanglement

From this, we can see that through entangling n sensing qubits, the **signal** we are trying to sense increases from $V_z \to nV_z$, and with $\frac{N}{n}$ total measurements,

$$\sigma_V \propto \frac{1}{n}\sigma_p \propto \frac{1}{n}\left(\frac{1}{\sqrt{\frac{N}{n}}}\right) = \frac{1}{\sqrt{Nn}}$$

which means the error decreased by a factor of $\sqrt{n}$. In the case where $n = N$, the uncertainty now scales as

$$\sigma_V \propto \frac{1}{N}$$

which is known as the Heisenberg limit, and is the quantum-mechanically limited, maximal amount of information one can get from taking n quantum sensing measurements, see again Giovannetti *et al.*, at https://www.nature.com/articles/nphoton.2011.35.

Two-qubit program example (best seen using the jupyter-notebook)

```
import numpy as np
from scipy.linalg import expm
import matplotlib.pyplot as plt

# Define single-qubit operations, Identity, Pauli, Hadamard and S matr
Id = np.array([[1, 0], [0, 1]], dtype=complex)
X = np.array([[0, 1], [1, 0]], dtype=complex)
Y = np.array([[0, -1j], [1j, 0]], dtype=complex)
Z = np.array([[1, 0], [0, -1]], dtype=complex)
Had = np.array([[1, 1],[1, -1]], dtype=complex) / np.sqrt(2)
S = np.array([[1, 0],[0, 1j]], dtype=complex)
# Define two-qubit gates
CNOT01 = np.array([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 0, 1], [0, 0, 1,
CNOT10 = np.array([[1, 0, 0, 0], [0, 0, 0, 1], [0, 0, 1, 0], [0, 1, 0,
SWAP = np.array([[1, 0, 0, 0], [0, 0, 1, 0], [0, 1, 0, 0], [0, 0, 0, 1

times = np.linspace(0, 1, 100)  # Time range from 0 to 1 seconds

# Initial state for each qubit
psi_1 =  np.array([1, 0], dtype=complex) # start with |0> for qubits 1
psi_2 =  np.array([1, 0], dtype=complex) # start with |0> for qubits 1
# possible basis states for measurements
basis_00 = np.array([1, 0, 0, 0], dtype=complex)
basis_01 = np.array([0, 1, 0, 0], dtype=complex)
basis_10 = np.array([0, 0, 1, 0], dtype=complex)
basis_11 = np.array([0, 0, 0, 1], dtype=complex)
```

Analysis and additions

The result here is pretty boring, we just get 0.5, as expected. The field acts only in the z-direction and is a constant. The probabilities are thus left unchanged. Feel free to change the Hamiltonian. Here's a list of suggestion:

1. More realistic Hamiltonian (keep in mind that if you make it time-dependent, you need to integrate over time. Different approximation exist.
2. Change the measurements to say making a measurement on only one of the qubits
3. More realistic readout functions and series of measurements as experiments
4. Study different entangled states
5. Find the field which acts on the system! How wuld you do that?
6. Study the Fisher entropy and much more

Conclusion

The readout functions in quantum sensing serve as the crucial step in extracting classical information from a quantum system after a set of operations. In quantum magnetometry, for instance, the readout is typically performed by measuring the qubit in the computational basis and using the outcome (such as the probabilities of measuring $|0\rangle$ or $|1\rangle$) to infer properties like the magnetic field strength. In more complex systems, the readout functions could involve more sophisticated measurements such as those based on quantum tomography, where the state of the system is fully reconstructed from measurement data.

Other types of sensors

This process can be generalized to other types of quantum sensors, such as those measuring electric fields, temperature, or time. The general concept remains: we manipulate a quantum system, measure it, and extract classical information to sense the desired physical quantity.
The examples below illustrate the diversity of quantum sensing applications, where quantum states (entanglement, superposition, squeezing) or quantum systems (atoms, spins, photons) enable breakthroughs in precision beyond classical limits.

Magnetic Field Sensing with Nitrogen-Vacancy (NV) Centers

- Parameter: Magnetic field strength.

Method/platform: NV centers in diamond have electron spins sensitive to magnetic fields. By optically detecting spin state changes (via fluorescence), magnetic fields at the nanoscale are estimated with high spatial resolution, useful in material science and biomedical imaging.

Atomic Clocks

- Parameter: Time/frequency.

Method/platform: Atomic transitions (e.g., in cesium or rubidium) serve as frequency standards. Quantum superposition states are probed to lock oscillator frequencies, enabling ultra-precise timekeeping critical for GPS and telecommunications.

Gravitational Wave Detection (LIGO)

- Parameter: Phase shift induced by spacetime ripples.

Method/platform: Squeezed light reduces quantum noise in interferometers, enhancing sensitivity to minute phase shifts caused by gravitational waves, pushing measurements below the standard quantum limit.

Quantum Thermometry

- Parameter: Temperature.

Method/platform**: Quantum probes like trapped ions or superconducting qubits exploit temperature-dependent decoherence or energy-level shifts to measure microkelvin-scale temperatures in cryogenic systems.

Entangled Photon Interferometry

- Parameter: Optical phase shifts.

Method/platform: Entangled photons in interferometers achieve sub-shot-noise precision, enabling enhanced measurements of distances or refractive indices for applications in metrology and imaging.

Electric Field Sensing with Rydberg Atoms

- Parameter: Electric field strength.

Method/platform: Rydberg atoms, highly sensitive to electric fields due to their large polarizability, detect field-induced Stark shifts via microwave spectroscopy, useful in electrometry and communications.

Quantum Gyroscopes

- Parameter: Rotation rate.

Method/platform: Cold atom interferometers or entangled particles exploit the Sagnac effect to measure rotation with quantum-enhanced precision, advancing inertial navigation systems.

Molecular Concentration Sensing

- Parameter: Chemical concentration.

Method/platform: Spin defects in diamond (e.g., NV centers) detect local magnetic perturbations from target molecules, enabling nanoscale NMR spectroscopy for biological or chemical analysis.

Superconducting Quantum Interference Devices (SQUIDs)

- Parameter: Magnetic flux.

Method/platform: SQUIDs leverage flux quantization and Josephson junctions to measure extremely weak magnetic fields, applied in magnetoencephalography (MEG) for brain activity mapping.

A more refined example

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import expm

class QuantumSensor:
    def __init__(self, shots=1000, time_steps=100):
        self.shots = shots
        self.time_steps = time_steps
        self.state = self.create_bell_state()
        self.measurement_results = np.zeros(shots)
        self.time_points = np.linspace(0, 10, self.time_steps)
        # B is a 3xN array where N is time_steps. B[:, t] gives the ma
        self.B = np.array([np.cos(self.time_points), np.sin(self.time_

    @staticmethod
    def create_bell_state():
        """Create an entangled Bell state (|00> + |11>)/sqrt(2)."""
        # Representing states in the computational basis |00>, |01>, |
        return (1/np.sqrt(2)) * (np.array([[1], [0], [0], [0]]) +
                                 np.array([[0], [0], [0], [1]]))

    @staticmethod
    def hamiltonian(B):
        """Create the Hamiltonian for the current magnetic field for a
        hbar = 1  # Planck's constant
        X = np.array([[0, 1], [1, 0]], dtype=complex)
        Y = np.array([[0, -1j], [1j, 0]], dtype=complex)
        Z = np.array([[1, 0], [0, -1]], dtype=complex)
```