This is a Linear Equation

$$A|x\rangle = |b\rangle$$

Unknown

Useful, READ VERY USEFUL

from communications to reunions

Solving a Non-Hermitian Linear System by Simulation

$$A|x\rangle = |b\rangle$$

of HHL Algorithm (in 2-dim)

Spoiler Alert: It is faster than classical in some ways

A A Gandhi (20319) Rehan(220502) Dr Ankur Raina -ECS417,IISERB

Introduction to the Problem Statement

The problem of solving differential equations has its applications in almost every science field known to man. As these systems of equations become large, solving them pushes the capabilities of classical computers.

In this project, we explored quantum computation and how it is used to solve systems of linear equations.

We take the Matrix Form of the Differential Equations in the following way:

$$A\,|x\rangle = |b\rangle$$

Where:- A: a N x N Hermitian matrix
B: some N-dimensional vector with unit length

Problem Statement:- Given A and B, we need to solve the above equation for the vector $|x\rangle$.

Now, we have the the problem in the Quantum Linear System form.

The Theory

Spectral Decomposition

$$A = \sum_i \lambda_i |v_i\rangle\langle v_i| \qquad A^{-1} = \sum_i \frac{1}{\lambda_i} |v_i\rangle\langle v_i|$$

v_i → eigenvectors

λ_i → eigenvalues

$$|b\rangle = \sum_i \beta_i |v_i\rangle$$

$$A|x\rangle = |b\rangle \Rightarrow |x\rangle = A^{-1}|b\rangle$$

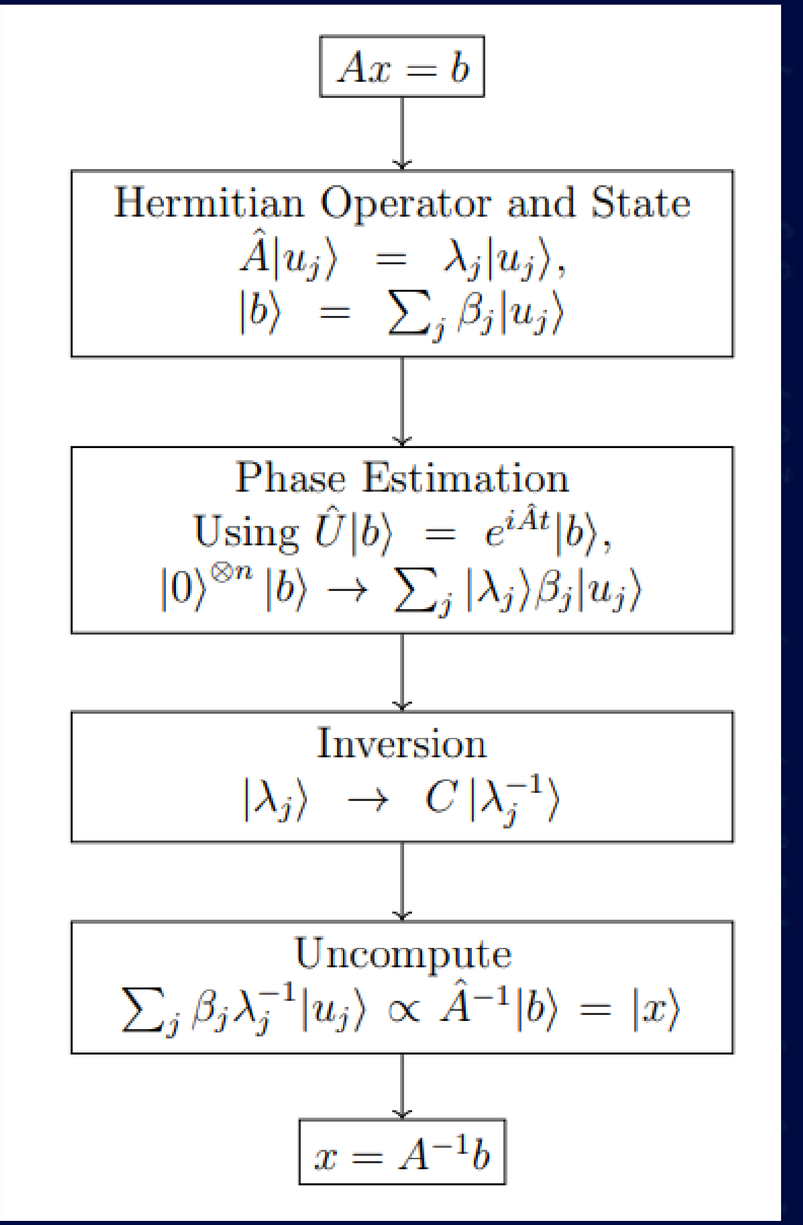
$$\left\{ |x\rangle = \sum_i \frac{\beta_i}{\lambda_i} |v_i\rangle \right\}$$

On the face of it we do not need eigenvalues or eigenvectors -----

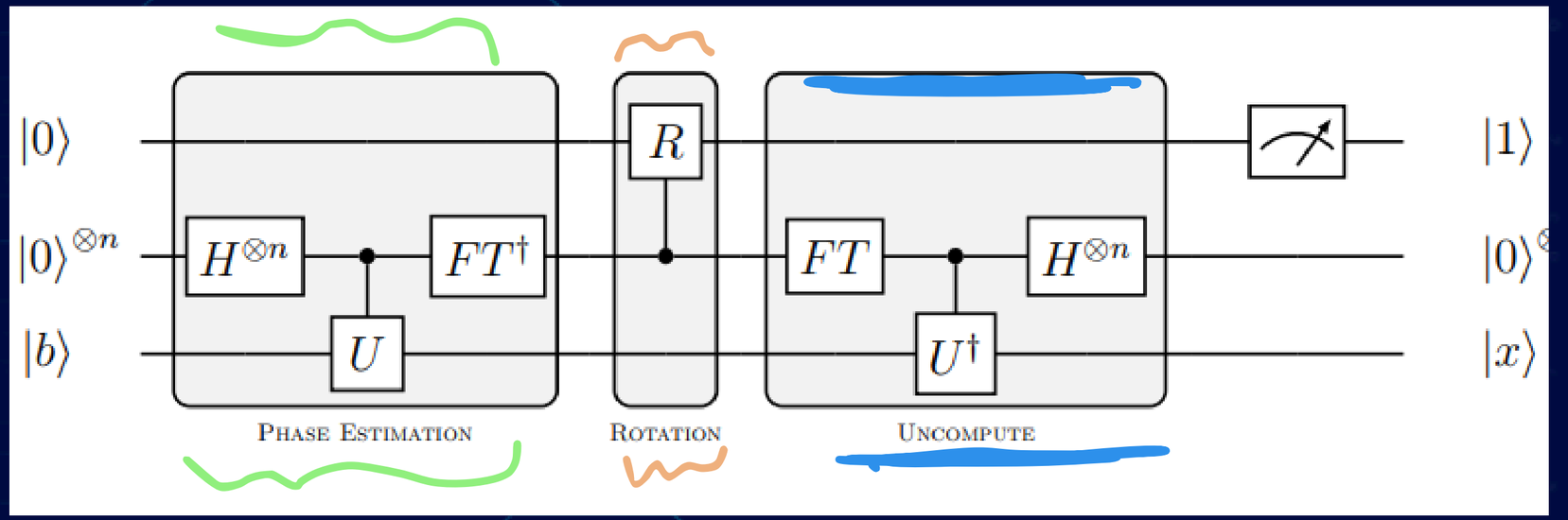
But life is more Complicated

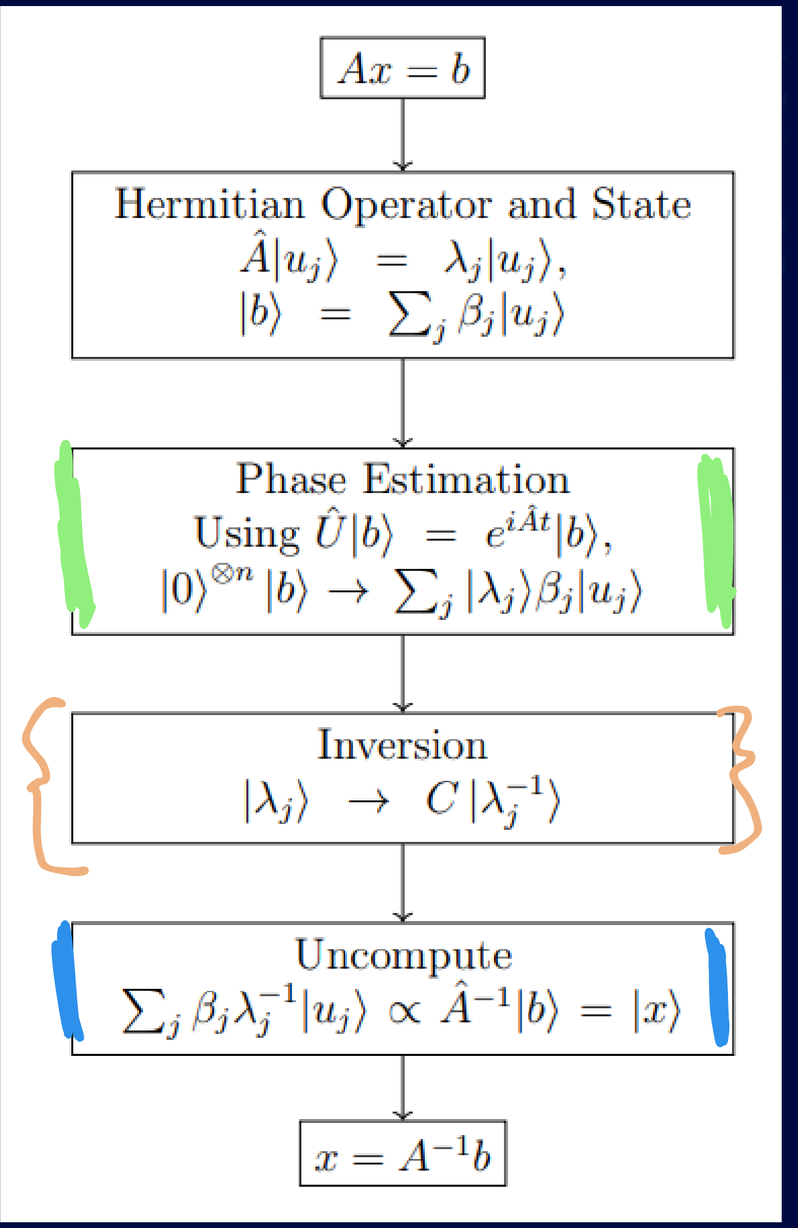
Birds Eye View

1. Conversion of the Differential Equation into the Quantum Linear System Equation (QLSE) form.
2. Getting the Hermitian Operator and State
3. Phase Estimation
4. Inversion
5. Uncomputing to get |x>



Birds Eye View





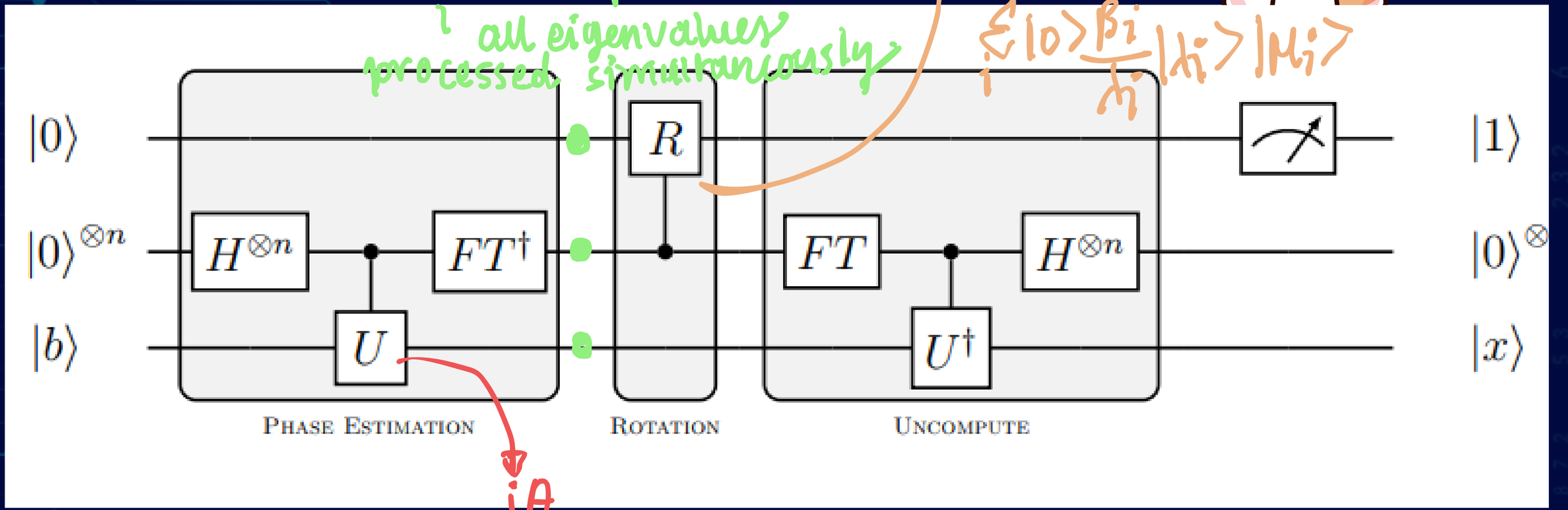
Quantum Speeding

$\sum_i |0\rangle \beta_i |\lambda_i\rangle |\mu_i\rangle$

errors are induced here

all eigenvalues processed simultaneously

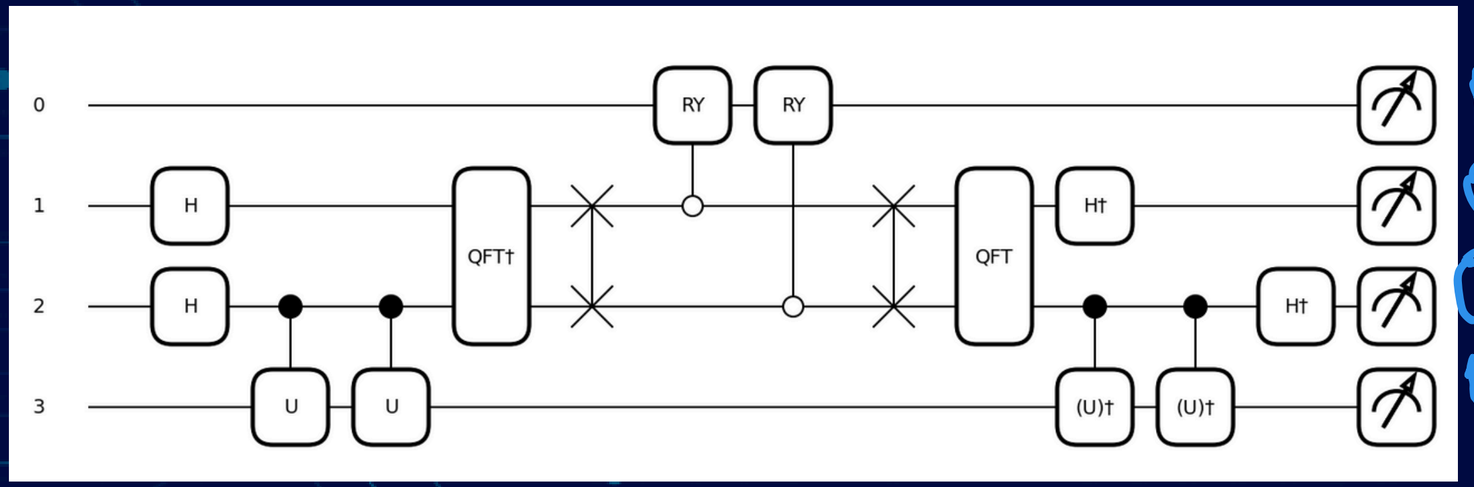
$\sum_i |0\rangle \frac{\beta_i}{\lambda_i} |\lambda_i\rangle |\mu_i\rangle$



e^{iA}

Finding this oracle can be a headache

Implementation



$$\begin{pmatrix} 3/2 & 1/2 \\ 1/2 & 3/2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.75 \\ -0.25 \end{pmatrix}$$

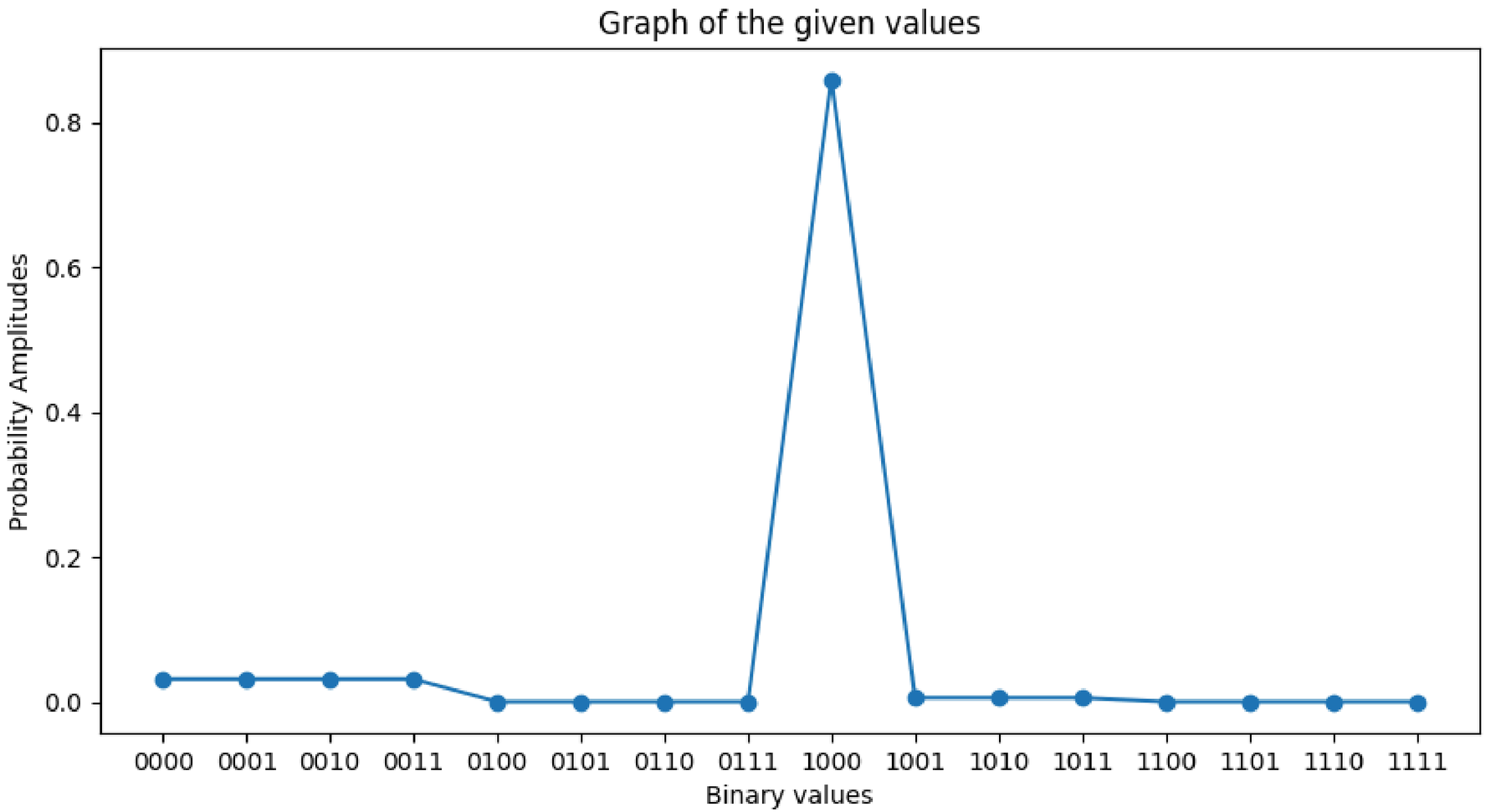
$$P(|1000\rangle) = 0.858$$

$$P(|1001\rangle) = 0.053$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.93 \\ 0.25 \end{pmatrix}$$

```python
import numpy as np
import pennylane as qml
import matplotlib.pyplot as plt



dev=qml.device("default.qubit",wires=4)

U1=np.array([[-1+1j,-1-1j],[-1-1j,-1+1j]])*(0.5)
U2=np.array([[0,1],[1,0]])
@qml.qnode(dev)
def Part1():
    #print(qml.state(wires=[0,1,2,3]))

    qml.Hadamard(wires=1)
    qml.Hadamard(wires=2)
    qml.ctrl(qml.QubitUnitary(U1,wires=3),2,1)
    qml.ctrl(qml.QubitUnitary(U2,wires=3),2,1)
    qml.adjoint(qml.QFT)(wires=[1,2])


    qml.SWAP(wires=[1,2])
    qml.ctrl(qml.RY,1,0)((np.pi),wires=0)
    qml.ctrl(qml.RY,2,0)((np.pi/2),wires=0)
    qml.SWAP(wires=[1,2])

    qml.QFT(wires=[1,2])
    qml.adjoint(qml.ctrl(qml.QubitUnitary(U2,wires=3),2,1))
    qml.adjoint(qml.ctrl(qml.QubitUnitary(U1,wires=3),2,1))
    qml.adjoint(qml.Hadamard)(wires=2)
    qml.adjoint(qml.Hadamard)(wires=1)
    #m_0 = qml.measure(wires=0)
    return qml.probs(wires=range(4))


c=Part1()
print(c)
fig, ax = qml.draw_mpl(Part1)()
fig.show()
```

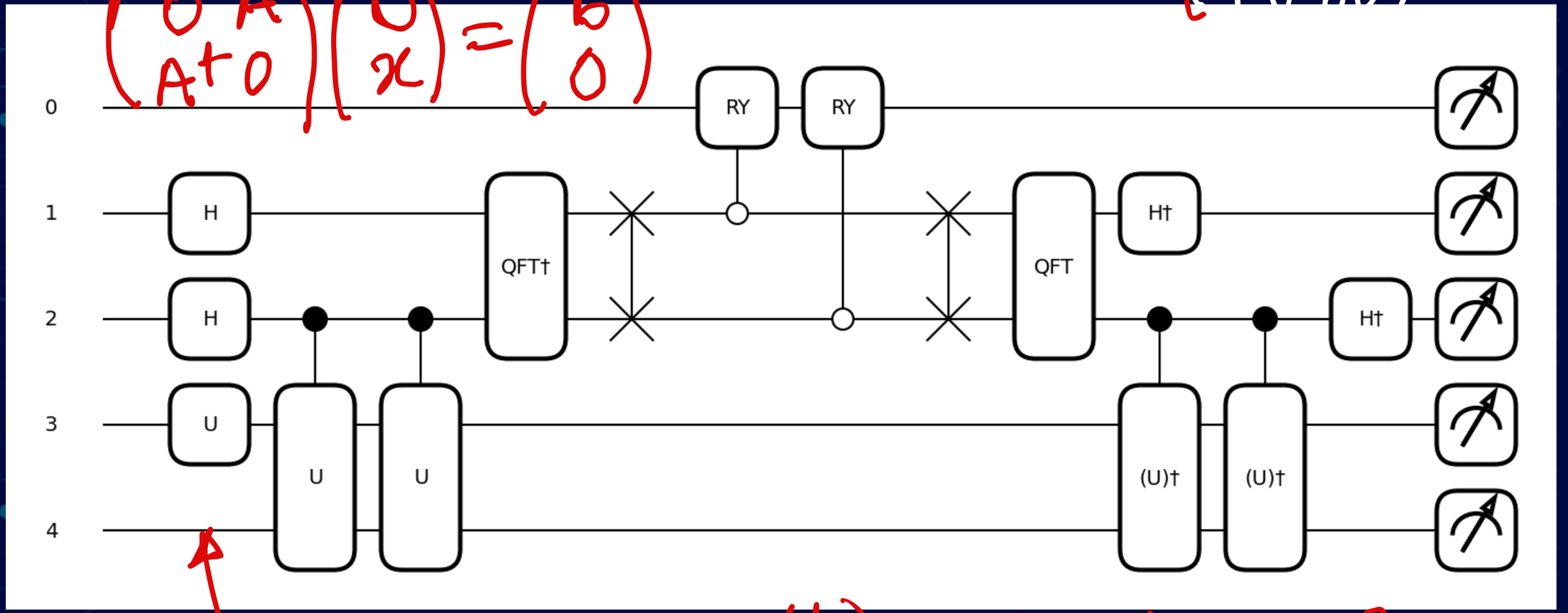
Results


Graph of the given values
Probability Amplitudes
0.0
0.2
0.4
0.6
0.8
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
Binary values

Our Modifications

$$A' = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} = \begin{pmatrix} 00 & 01 \\ 00 & 20 \\ 02 & 00 \\ 10 & 00 \end{pmatrix}$$

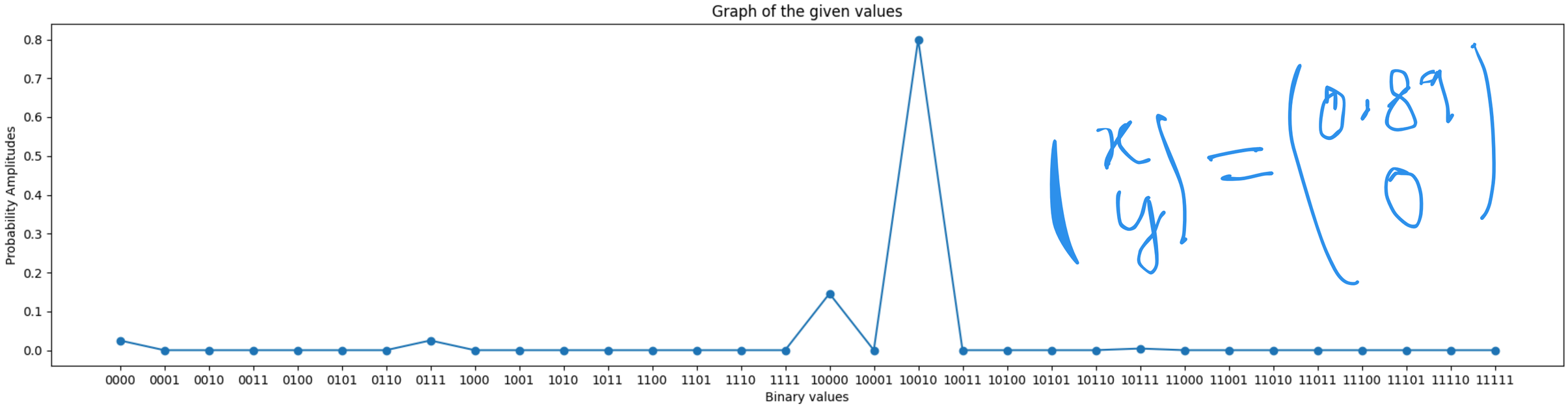
$$\begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \begin{pmatrix} 0 \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$



Puts the state in $\begin{pmatrix} b \\ 0 \end{pmatrix}$

$$b = \frac{1}{\sqrt{5}}|0\rangle + \frac{2}{\sqrt{5}}|1\rangle$$

Results



$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.89 \\ 0 \end{pmatrix}$

$P(|10010\rangle) = 0.8$

$P(|10011\rangle) \approx 0.$

$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

APPLICATIONS OF USING QC FOR SOLVING MULTI-QUBIT DIFFERENTIAL EQUATIONS

Machine Learning and AI

Data Search in Unstructured Data

Traffic and Transportation

Cryptography and Security

Optimization Problems

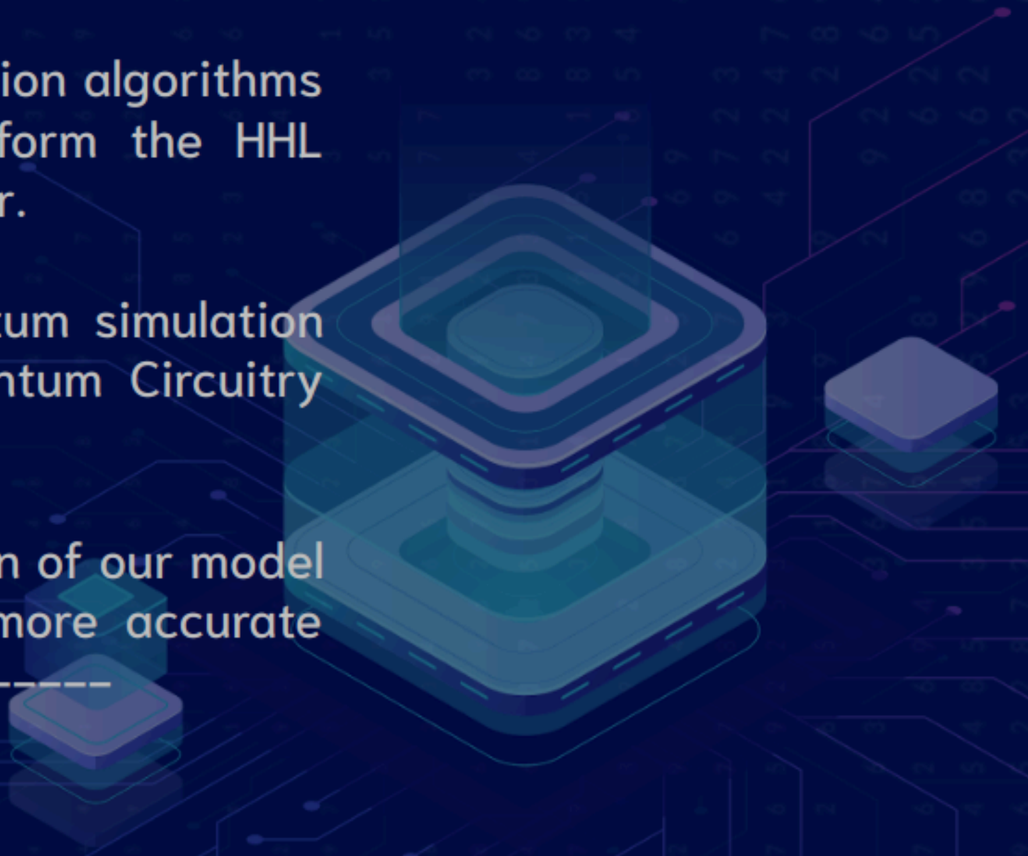
CONCLUSION

In this project presentation, we explored quantum computation and how it is used to solve systems of linear equations.

We used the Quantum Fourier Transform and Phase Estimation algorithms which in combination with the Uncomputing algorithm form the HHL algorithm for solving the linear systems on quantum computer.

We implemented the HHL algorithm with the use of quantum simulation softwares Quirk and Python Library Pennylane, for Quantum Circuitry Computation.

We then took a step further by extending the implementation of our model on higher frequency of qubit lines, which assures of a more accurate probability due to an increased sample set size.——————————



Challenges or Opportunities

- Demonstrated superior Efficiency but only for sparse matrices
- Limited to certain kind of matrices
- Answers are constant multiples
- Rotation is more difficult to be implemented
- Mostly Eigenvalues and eigenvectors should be known to have superior efficieny

However this is a demonstration that there might be better ways to Solve Linear Equation

References

- HHL Analysis and Simulation Verification Based on Origin Quantum Platform
by Xiaonan Liu1 , Lina Jing , Lin Han , and Jie Gao1

- USING QUANTUM ALGORITHMS TO SOLVE LINEAR SYSTEMS
by Michael C.R. Byrd Jr.

THANK YOU