Summer Q Camp 2019

Overview

Track 1 - Single Qubits

↓

Track 2 - Entanglement and multi-qubit systems

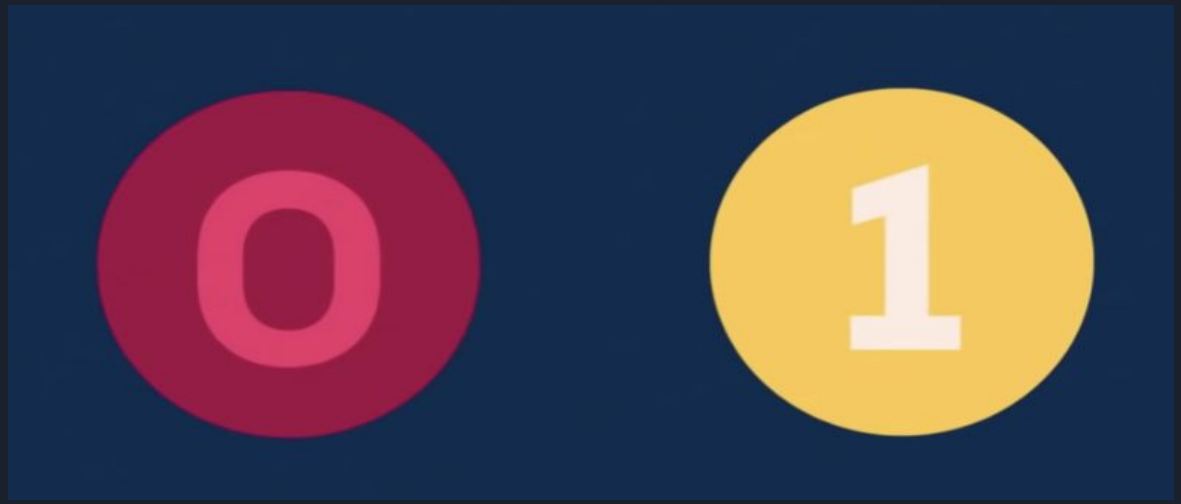
↓

Track 3 - State and process tomography

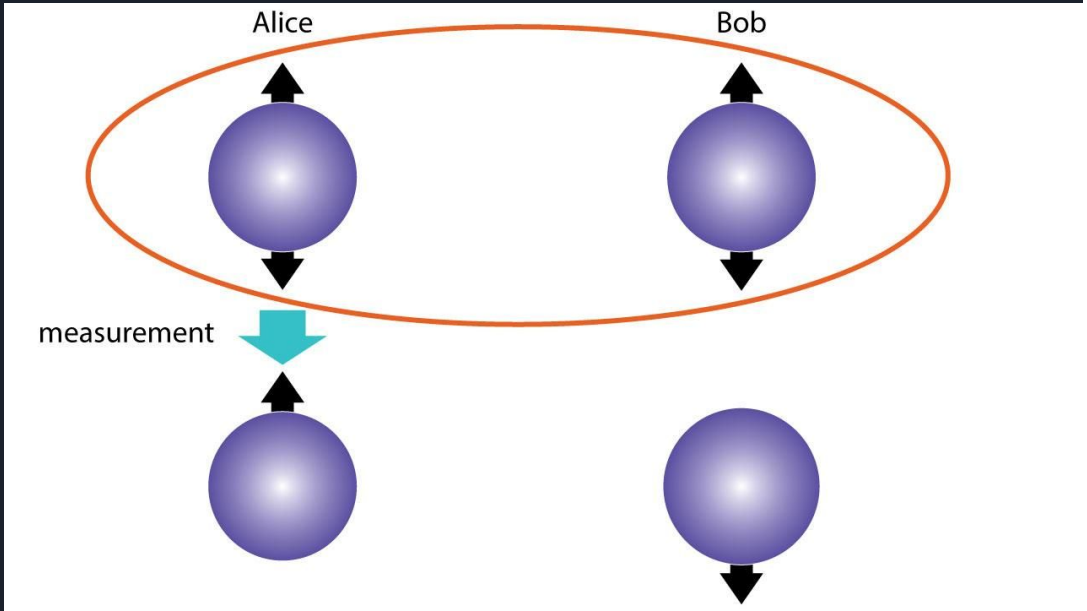
↓

Track 4 - Noise

Track 1 - Single Qubits



Track 2 - Multi-qubit systems and entanglement



$$|sing\rangle = \frac{1}{\sqrt{2}}(|ud\rangle - |du\rangle)$$

$$|T_1\rangle = \frac{1}{\sqrt{2}}(|ud\rangle + |du\rangle)$$

$$|T_2\rangle = \frac{1}{\sqrt{2}}(|uu\rangle + |dd\rangle)$$

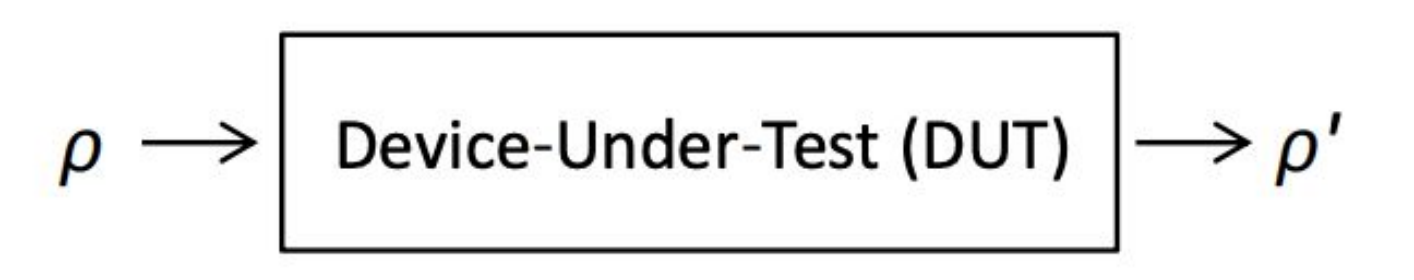
$$|T_3\rangle = \frac{1}{\sqrt{2}}(|uu\rangle - |dd\rangle)$$

Track 3 - QST and QPT

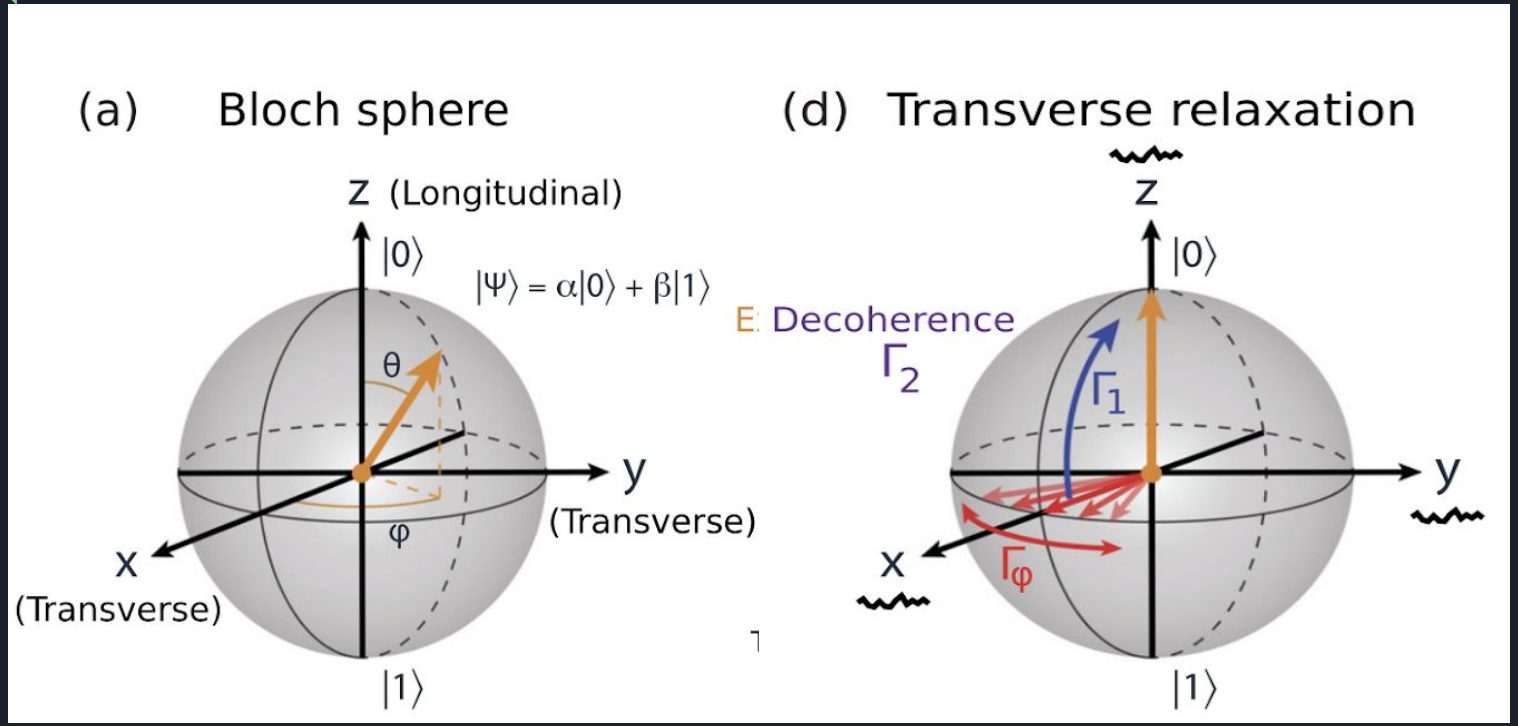
Quantum state tomography

$$\rho = \sum_{\vec{v}} \frac{\mathrm{tr}\left(\sigma_{v_1} \otimes \sigma_{v_2} \otimes \cdots \otimes \sigma_{v_n} \rho\right) \sigma_{v_1} \otimes \sigma_{v_2} \otimes \cdots \otimes \sigma_{v_n}}{2^n}$$

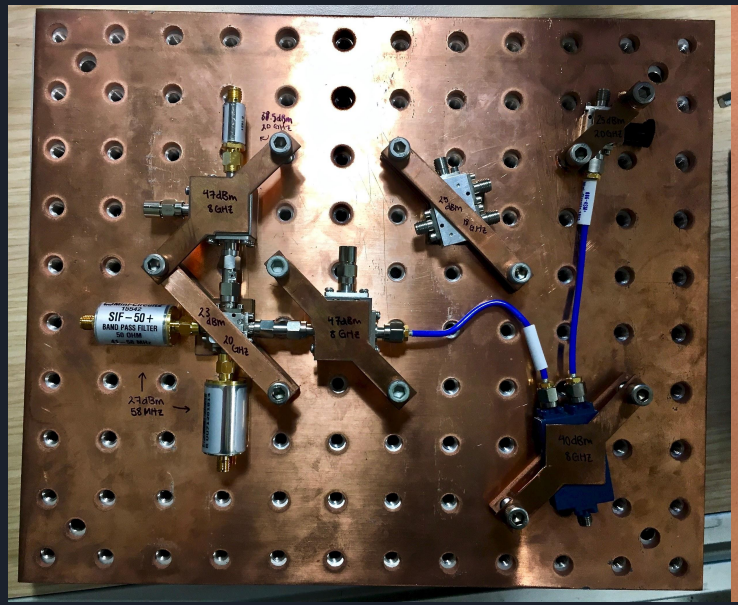
Quantum process tomography

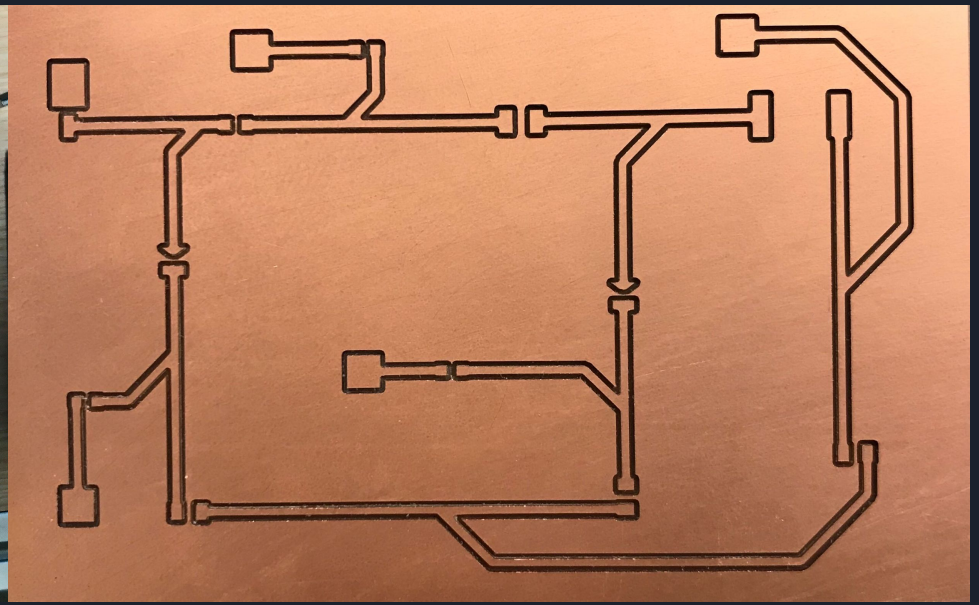


Track 4 - Noise



Hardware





TRACK 1

Maheshi
Eva

Qubits

Fundamental building blocks for quantum information processes.

Qubits can be in the states |0>, |1> or in a superposition state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where

$|\alpha|^2 + |\beta|^2 = 1$

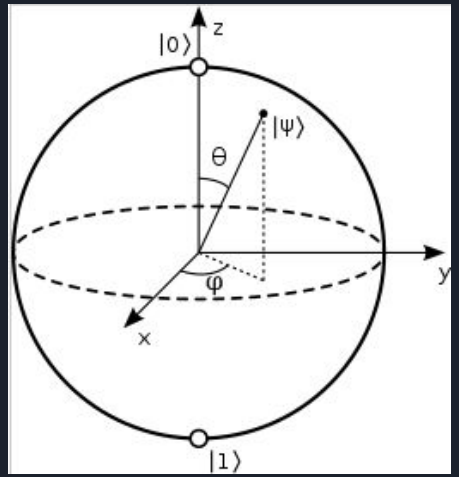
Dirac Notation:

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

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

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Bloch Sphere:



Gates

A quantum gate or process captures the dynamic change to a state which occurs as a result of some physical process.

Examples of Gates:

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

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

Gates in Action:

$$X|0\rangle = |1\rangle$$

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

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

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

Measurement

- Measurements in quantum mechanics are projective.
- Qiskit has a fixed measurement basis.
- Changing the measurement basis is a matter of perspective:

Example:

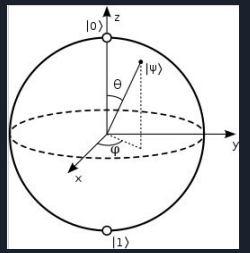
Basis: $\{|\uparrow\rangle, |\downarrow\rangle\}$

$$|\rightarrow\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle)$$

Basis: $\{|\rightarrow\rangle, |\leftarrow\rangle\}$

$$|\uparrow\rangle = \frac{1}{\sqrt{2}}(|\rightarrow\rangle + |\leftarrow\rangle)$$

How to run an experiment in Qiskit

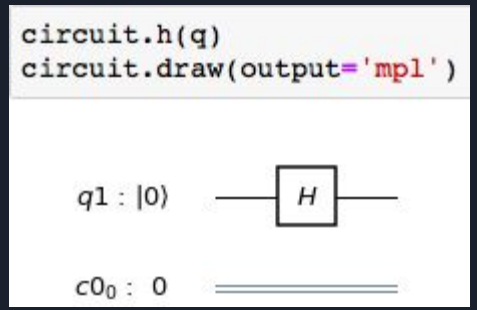


1. Set up circuit

```
q = QuantumRegister (1)
c = ClassicalRegister (1)

circuit = QuantumCircuit(q,c)
```

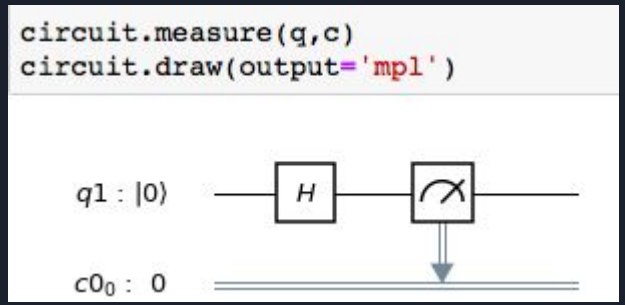
2. Add in a gate

```
circuit.h(q)
circuit.draw(output='mpl')
```



3. Add in circuit measurement

```
circuit.measure(q,c)
circuit.draw(output='mpl')
```

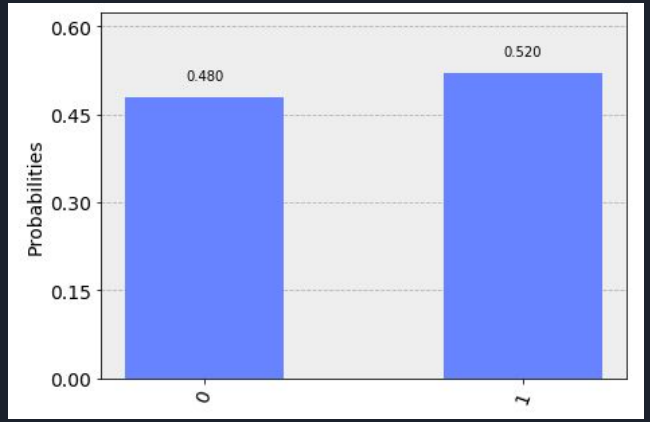


How to run an experiment in Qiskit

4.Determine count for circuit

```
job = execute(circuit, simulator, shots=100)
result = job.result()

counts = result.get_counts(circuit)
print("\nTotal count for 0 and 1 are:",counts)

Total count for 0 and 1 are: {'1': 52, '0': 48}
```

5.Put in a histogram



Rotation on x,y,z axis

```
qc = QuantumCircuit(q)
qc.rx(pi/2,q)
qc.draw(output='mpl')
```


```
job = execute(qc, backend)
```

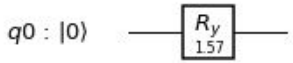
```
job.result().get_unitary(qc, decimals=3
```

```
array([[0.707+0.j   , 0.   -0.707j],
       [0.   -0.707j, 0.707+0.j   ]])
```

$$R_x(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} = u3(\theta, -\pi/2, \pi/2)$$

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2). \end{pmatrix} = u3(\theta, 0, 0)$$

```
qc = QuantumCircuit(q)
qc.ry(pi/2,q)
qc.draw(output='mpl')
```




```
job = execute(qc, backend)
job.result().get_unitary(qc, decimals=3)
```

```
array([[ 0.707+0.j, -0.707+0.j],
       [ 0.707+0.j,  0.707+0.j]])
```

```
qc = QuantumCircuit(q)
qc.rz(pi/2,q)
qc.draw(output='mpl')
```


```
job = execute(qc, backend)
job.result().get_unitary(qc, decimals=3)
```

```
array([[1.+0.j, 0.+0.j],
       [0.+0.j, 0.+1.j]])
```

$$R_z(\phi) = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix} \equiv u1(\phi)$$

TRACK 2

Elizabeth
Ariane

In a classical system, probability is simple

Uncorrelated

$$P_u(a,b) = P_A(a)P_B(b)$$

Correlated

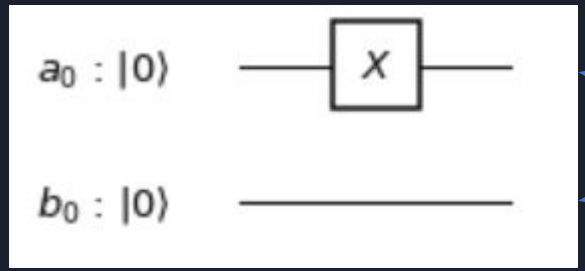
$$P_c(a,b) = P(a,b) - P_A(a)P_B(b)$$

In the uncorrelated case, A does not affect B at all while in the correlated case, it does.

Correlated: If A = 0.75, B must be 0.25

Product State

The product state on Qiskit is almost like a single qubit state



Only one qubit is met with a single qubit gate, the other is unaffected

Entangled States

$$|sing\rangle = \frac{1}{\sqrt{2}}(|ud\rangle - |du\rangle)$$

$$|T_1\rangle = \frac{1}{\sqrt{2}}(|ud\rangle + |du\rangle)$$

$$|T_2\rangle = \frac{1}{\sqrt{2}}(|uu\rangle + |dd\rangle)$$

$$|T_3\rangle = \frac{1}{\sqrt{2}}(|uu\rangle - |dd\rangle)$$

Expectation value - the average of all possible outcomes of a measurement.

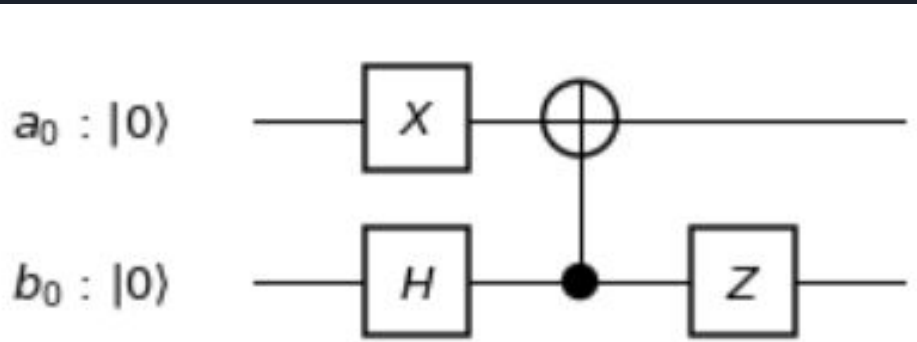
$$\begin{aligned}
\sigma_z |uu\rangle &= |uu\rangle \\
\sigma_z |du\rangle &= -|du\rangle \\
\sigma_x |ud\rangle &= |dd\rangle \\
\sigma_x |dd\rangle &= |ud\rangle \\
\sigma_y |uu\rangle &= i|du\rangle \\
\sigma_y |du\rangle &= -i|uu\rangle
\end{aligned}$$

$$\langle T_1 | \sigma_z \tau_z | T_1 \rangle = \langle T_1 | \frac{1}{\sqrt{2}} (|ud\rangle + |du\rangle)$$

If we take these equations and apply them to the triplet state, it can be simplified and we can determine the expectation value of the state.

Singlet State

```
# singlet state
qc.x(a)
qc.h(b)
qc.cx(b,a)
qc.z(b)
```

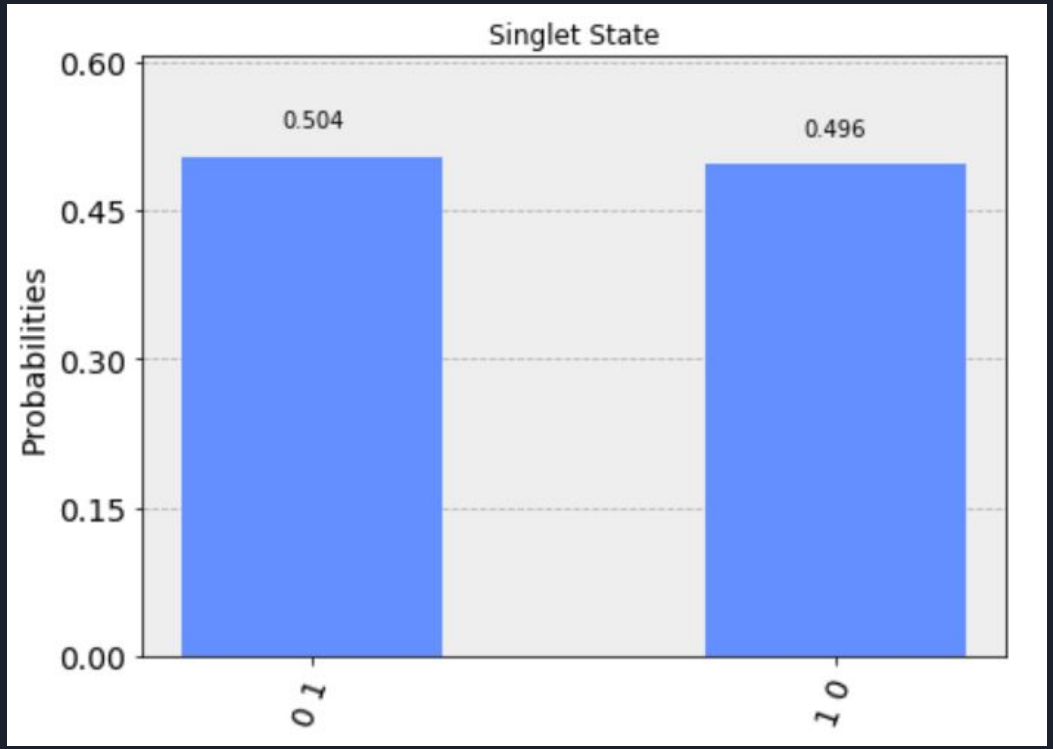



```
State Vector:
[ 0.         +0.j  0.70710678+0.j -0.70710678+0.j  0.         +0.j]
```

$$|sing\rangle = \frac{1}{\sqrt{2}}(|ud\rangle - |du\rangle)$$

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{bmatrix} = \begin{bmatrix} |dd\rangle \\ |du\rangle \\ |ud\rangle \\ |uu\rangle \end{bmatrix}$$

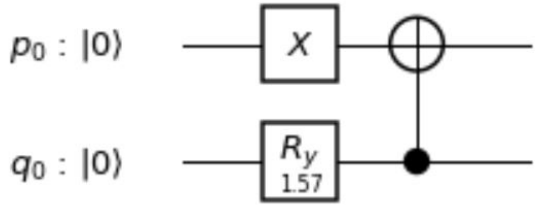
```
State Vector:
[ 0.          +0.j  0.70710678+0.j -0.70710678+0.j  0.          +0.j]
```



$$|sing\rangle = \frac{1}{\sqrt{2}}(|ud\rangle - |du\rangle)$$

Triplet States

```
# triplet state assignment 1
qc.x(p)
qc.ry(pi/2,q)
qc.cx(q,p)
```




```
State Vector:
[0.         +0.j 0.70710678+0.j 0.70710678+0.j 0.         +0.j]
```

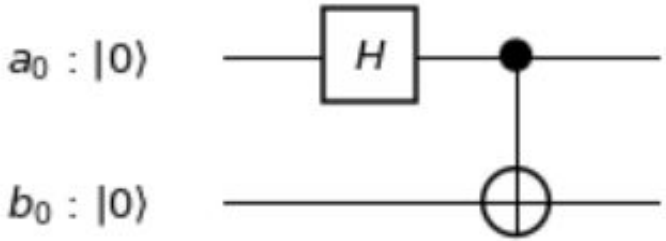
$$|T_1\rangle = \frac{1}{\sqrt{2}}\left(|ud\rangle + |du\rangle\right)$$

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{bmatrix} = \begin{bmatrix} |dd\rangle \\ |du\rangle \\ |ud\rangle \\ |uu\rangle \end{bmatrix}$$

```
State Vector:
[0.         +0.j 0.70710678+0.j 0.70710678+0.j 0.         +0.j]
```

Triplet States

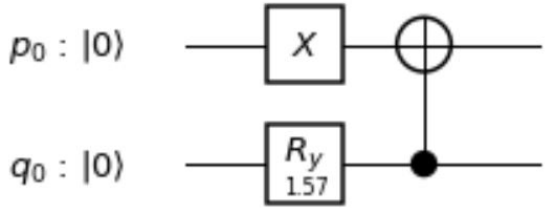
```
# triplet state assignmemt 2
qc.h(a)
qc.cx(a,b)
```




```
State Vector:
[0.70710678+0.j 0.          +0.j 0.          +0.j 0.70710678+0.j]
```

Triplet States

```
# triplet state assignment 3
qc.h(a)
qc.cx(a,b)
qc.z(b)
```




```
State Vector:
[ 0.70710678+0.j  0.        +0.j  0.        +0.j -0.70710678+0.j]
```

Expectation Value

Expectation Values

X: -1

Y: 1

Z: 1

$$\sigma_z \tau_z |T_1\rangle = \sigma_z \tau_z (\frac{1}{\sqrt{2}})(|ud\rangle + |du\rangle)$$
$$= \tau_z (\frac{1}{\sqrt{2}})(|ud\rangle - |du\rangle)$$
$$= (\frac{1}{\sqrt{2}})(-|ud\rangle - |du\rangle)$$
$$= \frac{1}{2}(|ud\rangle + |du\rangle)(-|ud\rangle - |du\rangle)$$
$$= \frac{1}{2}(-1+(-1))$$
$$= -1$$

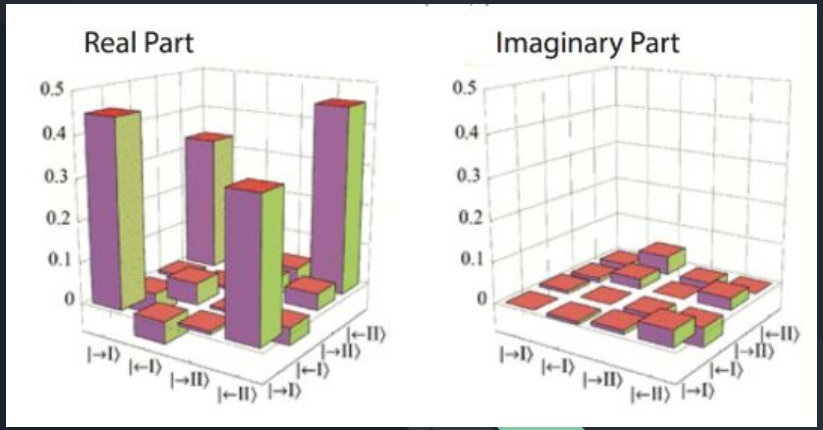
Expectation Value: -1

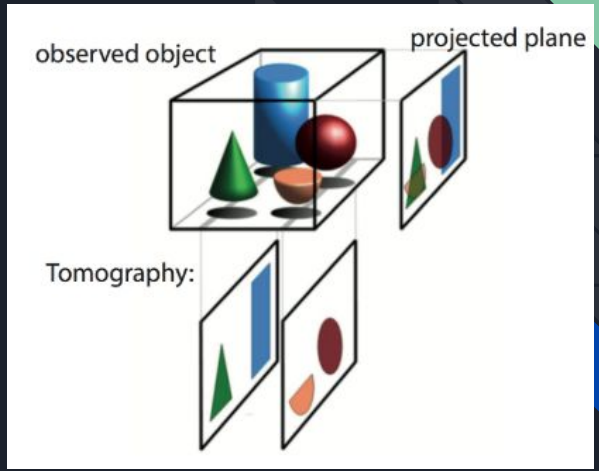
```
def expectationValue(qc):
    original = stater(qc)

    a = QuantumRegister(2)
    b = ClassicalRegister(2)

    # create a duplicate circuit flipped around the x axis (conjugate of the original)
    x_circuit = deepcopy(qc)
    x_circuit.x(0)
    x_circuit.x(1)

    # create a duplicate circuit flipped around the y axis (conjugate of the original)
    y_circuit = deepcopy(qc)
    y_circuit.y(0)
    y_circuit.y(1)

    # create a duplicate circuit flipped around the z axis (conjugate of the original)
    z_circuit = deepcopy(qc)
    z_circuit.z(0)
    z_circuit.z(1)
```

TRACK 3

Eva
Abigail

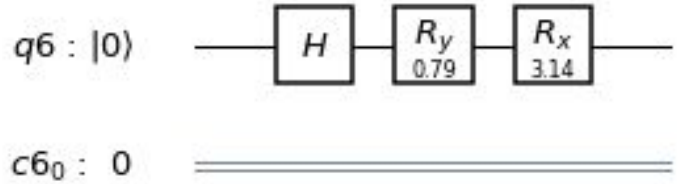
Quantum State Tomography

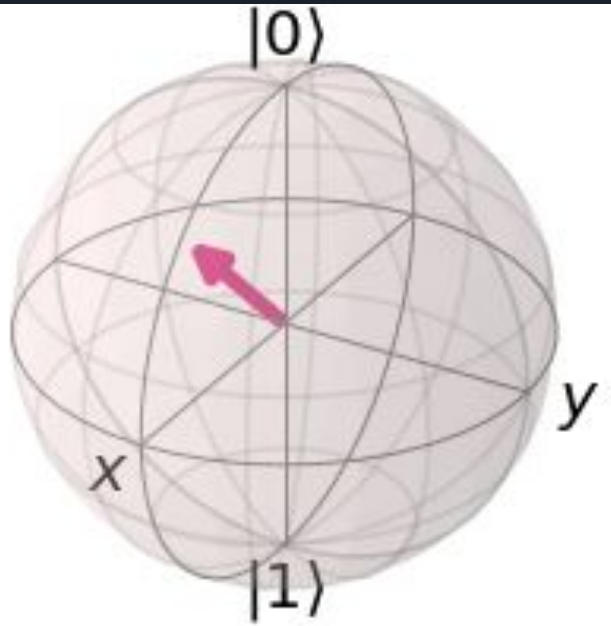




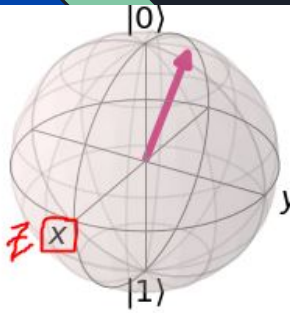
Prepare a State:

```
q = QuantumRegister(1)
c = ClassicalRegister(1)
circ = QuantumCircuit(q,c)
circ.h(q)
circ.ry(pi/4,q)
circ.rx(pi,q)
circ.draw(output='mpl')
```



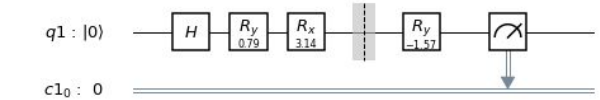


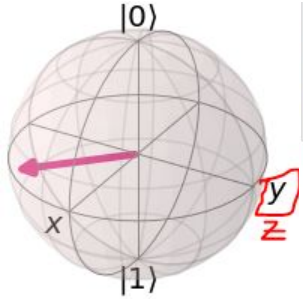
Perform QST




```
circX = deepcopy(circ)
circX.barrier()
circX.ry(-pi/2,q)
circY.barrier()
print(getCounts(circX,q,c,2000))
circX.draw(output = 'mpl')
```

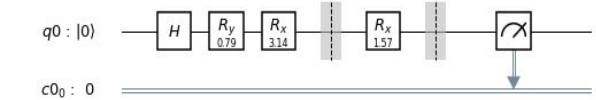
```
{'0': 1651, '1': 349}
```

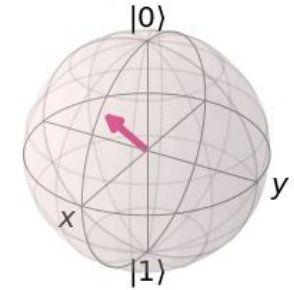





```
circY = deepcopy(circ)
circY.barrier()
circY.rx(pi/2,q)
circY.barrier()
print(getCounts(circY,q,c,2000))
circY.draw(output = 'mpl')
```

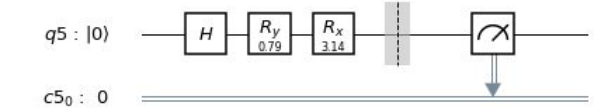
```
{'0': 986, '1': 1014}
```






```
circZ = deepcopy(circ)
circZ.barrier()
print(getCounts(circZ,q,c,2000))
circZ.draw(output = 'mpl')
```

```
{'0': 1712, '1': 288}
```



Finding X- Component:

$$\frac{1651-149}{2000} = 0.751$$

Finding Y- Component

$$\frac{986-1014}{2000} = -0.014$$

Finding Z- Component

$$\frac{1712-288}{2000} = 0.712$$

Density matrix representation for 1 qubit:

$$\rho = \frac{1}{2}(c_1\sigma_x + c_2\sigma_y + c_3\sigma_z + I)$$

```python
def getDensityMatrix(circuit,q,c):
    components = densityMatrixComponents(circuit,q,c, 50000)

    #Pauli operators are stored in a dictionary
    X = np.array([[0.+0.j, 1.+0.j], [1.+0.j, 0.+0.j]])
    Y = np.array([[0.+0.j, 0.-1.j], [0.+1.j, 0.+0.j]])
    Z = np.array([[1.+0.j, 0.+0.j], [0.+0.j, -1.+0.j]])
    I = np.array([[1.+0.j, 0.+0.j], [0.+0.j, 1.+0.j]])
    pauli = {'I': I, 'X': X, 'Y': Y, 'Z': Z}

    densityMatrix = 0
    for i,j in pauli.items():
        if q.size == 1:
            densityMatrix += j*components.get(i)
        else:
            for k,l in pauli.items():
                densityMatrix += np.kron(j, l)*components.get(i+k)

    print(densityMatrix*(1/2**(q.size)))
```

Our Result:

```python
#One-qubit test
q = QuantumRegister(1)
c = ClassicalRegister(1)
circ = QuantumCircuit(q,c)
circ.h(q)
circ.ry(pi/4,q)
circ.rx(pi,q)

getDensityMatrix(circ,q,c)
```

```
[[0.8535+0.j     0.3525+0.003j]
 [0.3525-0.003j 0.1465+0.j    ]]
```

Qiskit State Tomography:

```python
#Comparison with my getDensityMatrix Result
q1 = QuantumRegister(1)
c1 = ClassicalRegister(1)
circ1 = QuantumCircuit(q1,c1)
circ1.h(q1)
circ1.ry(np.pi/4,q1)
circ1.rx(np.pi,q1)

qst_bell = state_tomography_circuits(circ1, q1)
job = qiskit.execute(qst_bell, Aer.get_backend('qasm_simulator'), shots=5000)

tomo_bell = StateTomographyFitter(job.result(), qst_bell)

np.round_(tomo_bell.fit(),3)
```

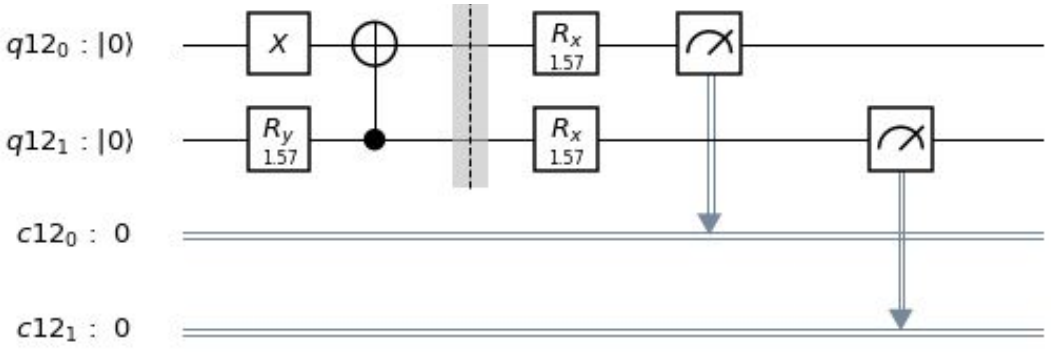
```
[[0.856+0.j    0.35 -0.019j]
 [0.35 +0.019j 0.144+0.j   ]]
```

Measurements for 2-qubits Example:

Example of measuring Y-components:

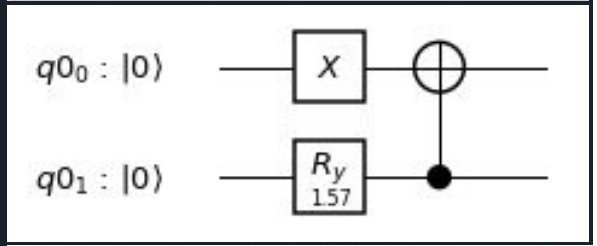
```
circuits[4].rx(pi/2,1)
circuits[4].rx(pi/2,0)
circuits[4].measure(q,c)
counts4 = getCounts(circuits[4],q,c,shots)
eValues['YI'] = expectationValue(counts4,'f', shots)
eValues['IY'] = expectationValue(counts4,'s', shots)
eValues['YY'] = expectationValue(counts4,'b', shots)
```

```
{'00': 4951, '11': 5049}
```



General Output of the function:

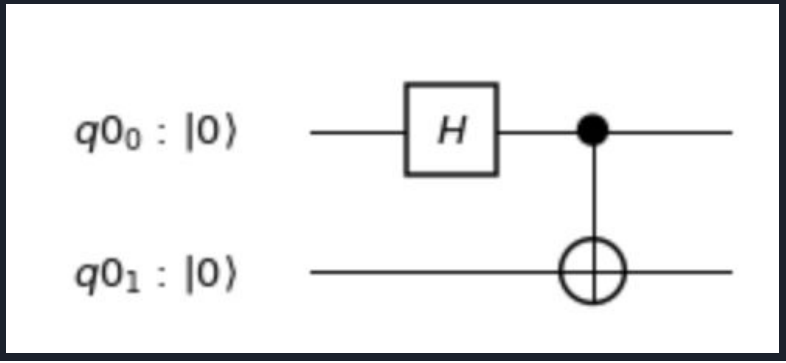
```
#Testing function on Triplet 0 state
q = QuantumRegister(2)
c = ClassicalRegister(2)
circT = QuantumCircuit(q,c)
circT.x(0)  #flip second qubit
circT.ry(pi/2, 1) #superposes first qubit
circT.cx(q[1],q[0])

densityMatrixComponents(circT,q,c, 10000)
```




```
{'XI': 0.009,
 'IX': 0.009,
 'XX': 1.0,
 'XY': 0.007,
 'XZ': 0.007,
 'YX': 0.008,
 'YI': 0.008,
 'IY': 0.008,
 'YY': 1.0,
 'YZ': 0.005,
 'ZX': 0.007,
 'ZY': 0.008,
 'ZI': 0.005,
 'IZ': -0.005,
 'ZZ': -1.0,
 'II': 1}
```

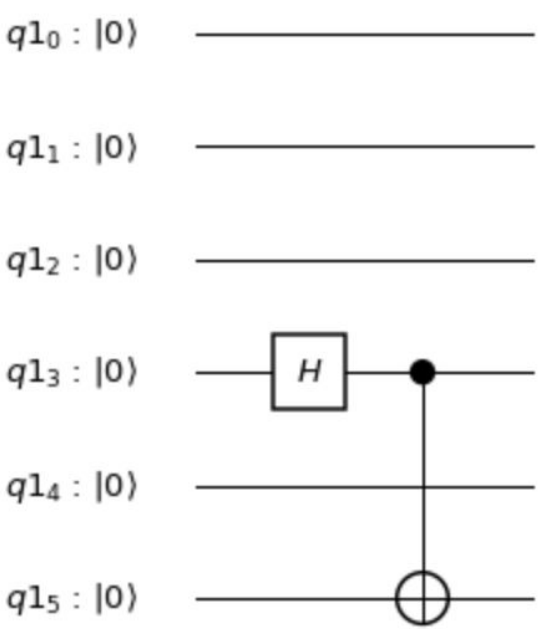
```
[[-0.   +0.j    -0.   +0.002j  0.   -0.002j  0.   +0.001j]
 [-0.   -0.002j  0.5  +0.j     0.5  -0.002j  0.002+0.002j]
 [ 0.   +0.002j  0.5  +0.002j  0.5  +0.j     0.002-0.001j]
 [ 0.   -0.001j  0.002-0.002j  0.002+0.001j  0.   +0.j   ]]
```

QISKIT Tomography




```
# Create the expected density matrix
q2 = QuantumRegister(2)
bell = QuantumCircuit(q2)
bell.h(q2[0])
bell.cx(q2[0], q2[1])

job = qiskit.execute(bell, Aer.get_backend('statevector_simulator'))
psi_bell = job.result().get_statevector(bell)
print(psi_bell)

bell.draw(output="mpl")
```

```
[0.70710678+0.j 0.        +0.j 0.        +0.j 0.70710678+0.j]
```




```
# Generate circuits and run on simulator
t = time.time()
# Generate the state tomography circuits. Only pass in the
# registers we want to measure (in this case 3 and 5)
qst_bell = state_tomography_circuits(bell, [q2[3],q2[5]])
job = qiskit.execute(qst_bell, Aer.get_backend('qasm_simulator'), shots=5000)
print('Time taken:', time.time() - t)

tomo_bell = StateTomographyFitter(job.result(), qst_bell)
```

```
Time taken: 0.3001821041107178
```

```
# Perform the tomography fit
# which outputs a density matrix
rho_bell = tomo_bell.fit()
F_bell = state_fidelity(psi_bell, rho_bell)
print('Fit Fidelity =', F_bell)
```

```
Fit Fidelity = 0.9999205355429333
```

Qiskit Density Matrix:

```
np.round_(rho_bell,2)
```

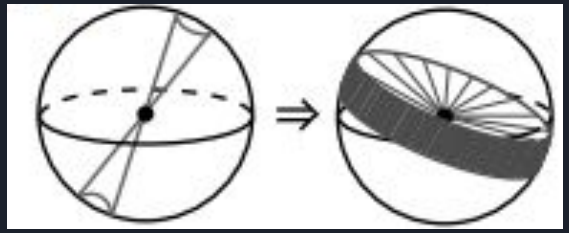
```
array([[ 0.5 +0.j  ,  0.  -0.j  , -0.  -0.j  ,  0.49-0.01j],
       [ 0.  +0.j  ,  0.  +0.j  , -0.  -0.j  ,  0.  +0.j  ],
       [-0.  +0.j  , -0.  +0.j  ,  0.  +0.j  , -0.  +0.j  ],
       [ 0.49+0.01j,  0.  -0.j  , -0.  -0.j  ,  0.5 +0.j  ]])
```

Expected:

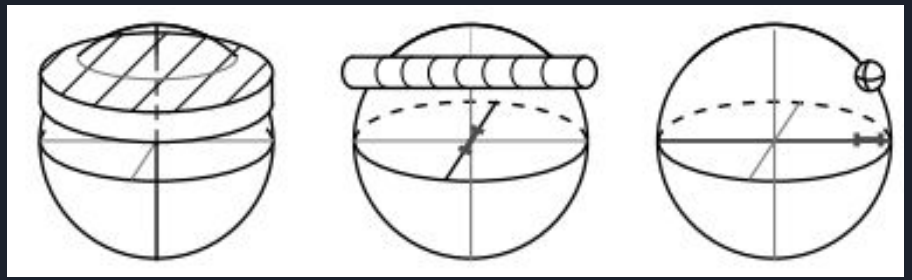
```
[[0.5+0.j 0. +0.j 0. +0.j 0.5+0.j]
 [0. +0.j 0. +0.j 0. +0.j 0. +0.j]
 [0. +0.j 0. +0.j 0. +0.j 0. +0.j]
 [0.5+0.j 0. +0.j 0. +0.j 0.5+0.j]]
```

Potential Errors in Deriving Density Matrix

- Errors in the measurement- basis vectors drift



- Errors from counting statistics - any real measurements can only be carried out with finite counts
- Errors in experimental stability



AND	The output is 1 only when both inputs are 1, otherwise the output is 0.		Input	Output
			0 0	0
			0 1	0
			1 0	0
			1 1	1

Quantum Process Tomography

How to do QPT - 1-Qubit Example

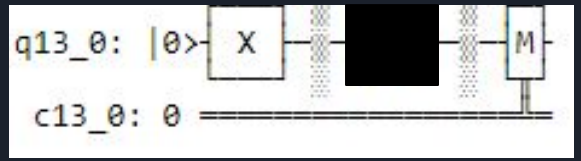
- We prepare the initial states Z+, Z-, X+, and Y+
  - + Z -> |0>
  - - Z -> |1>
  - +X -> (1/√2)(|0> + |1>)
  - +Y -> (1/√2)(|0> + *i*|1>)

Example: -Z

X Component

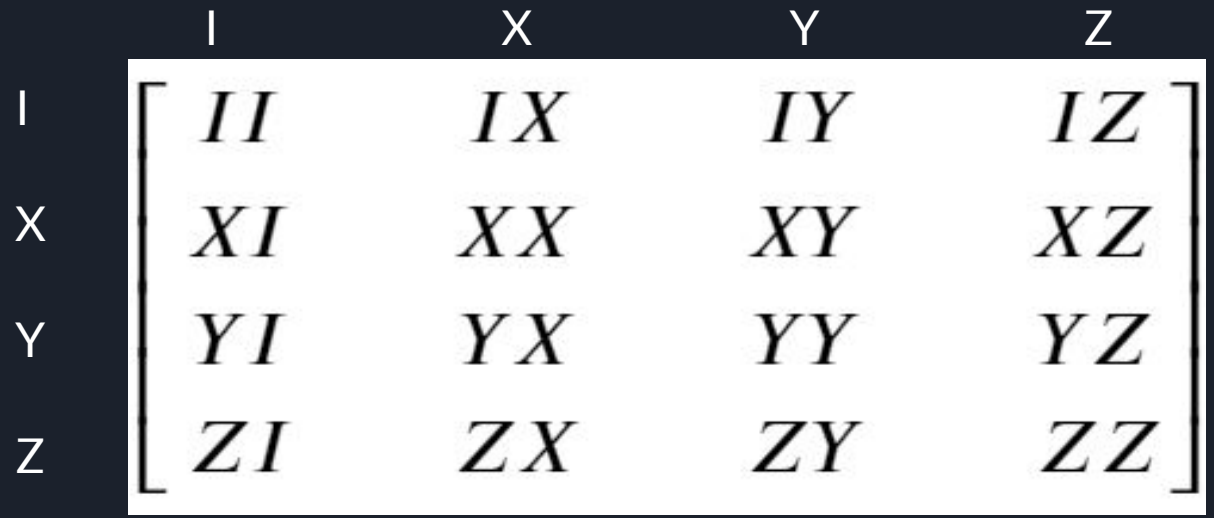
Y Component

Z Component



Characterizing the Process

- Can characterize the process using different matrix representations such as superoperator, Chi, Choi, etc.
- We use Pauli Transfer Matrix (PTM)

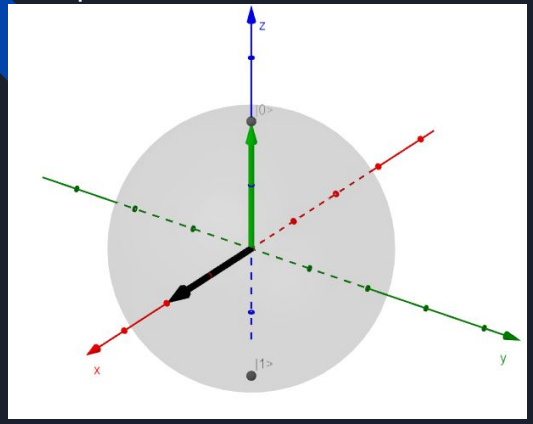


```
# Process tomography of a Hadamard gate
q = QuantumRegister(1)
circ = QuantumCircuit(q)
circ.h(q[0])

# Run circuit on unitary simulator to find ideal unitary
job = qiskit.execute(circ, Aer.get_backend('unitary_simulator'))
ideal_unitary = job.result().get_unitary(circ)
# convert to Choi-matrix in column-major convention
choi_ideal = Choi(outer(ideal_unitary.ravel(order='F')))
chi_ideal = Chi(choi_ideal)
ptm_ideal = PTM(choi_ideal)
superop_ideal = SuperOp(choi_ideal)
# Generate process tomography circuits and run on qasm simulator
qpt_circs = process_tomography_circuits(circ, q)
job = qiskit.execute(qpt_circs, Aer.get_backend('qasm_simulator'), shots=100)

# Extract tomography data so that counts are indexed by measurement configuration
qpt_tomo = ProcessTomographyFitter(job.result(), qpt_circs)
qpt_tomo.data
```

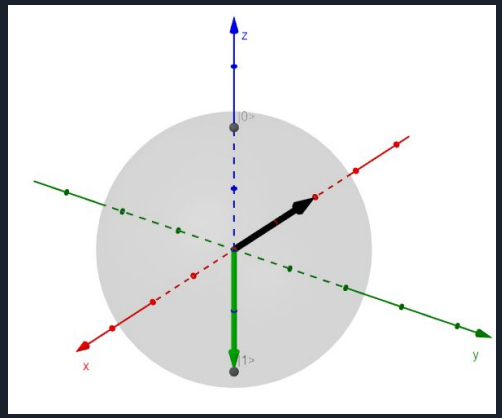
```
{(('Zp',), ('X',)): {'0': 100},
 (('Zp',), ('Y',)): {'1': 49, '0': 51},
 (('Zp',), ('Z',)): {'1': 43, '0': 57},
 (('Zm',), ('X',)): {'1': 100},
 (('Zm',), ('Y',)): {'1': 52, '0': 48},
 (('Zm',), ('Z',)): {'1': 47, '0': 53},
 (('Xp',), ('X',)): {'1': 47, '0': 53},
 (('Xp',), ('Y',)): {'1': 48, '0': 52},
 (('Xp',), ('Z',)): {'0': 100},
 (('Yp',), ('X',)): {'1': 39, '0': 61},
 (('Yp',), ('Y',)): {'1': 100},
 (('Yp',), ('Z',)): {'1': 50, '0': 50}}
```

Results:

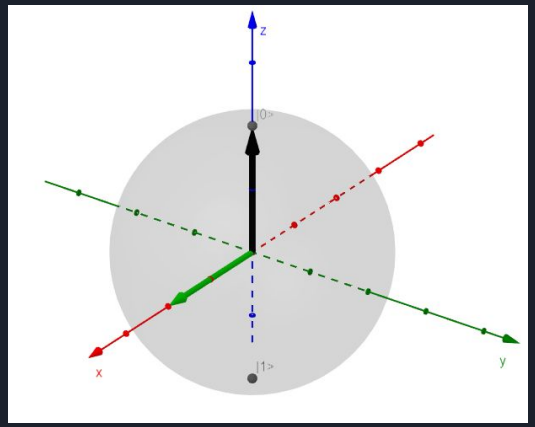
For |0>



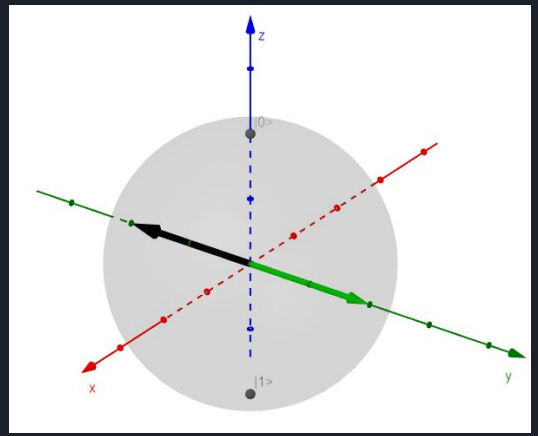
For |1>



For (1/√2)(|0> + |1>)



For (1/√2)(|0> + *i*|1>)




```
{(('Zp',), ('X',)): {'0': 4000},
 (('Zp',), ('Y',)): {'0': 2033, '1': 1967},
 (('Zp',), ('Z',)): {'0': 1985, '1': 2015},
 (('Zm',), ('X',)): {'1': 4000},
 (('Zm',), ('Y',)): {'0': 1959, '1': 2041},
 (('Zm',), ('Z',)): {'0': 2026, '1': 1974},
 (('Xp',), ('X',)): {'0': 2030, '1': 1970},
 (('Xp',), ('Y',)): {'0': 1993, '1': 2007},
 (('Xp',), ('Z',)): {'0': 4000},
 (('Yp',), ('X',)): {'0': 2040, '1': 1960},
 (('Yp',), ('Y',)): {'1': 4000},
 (('Yp',), ('Z',)): {'0': 1979, '1': 2021}}
```

PTM Representation:

$$\begin{array}{c} \\ I \\ X \\ Y \\ Z \end{array} \begin{array}{c} \begin{array}{cccc} I & X & Y & Z \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

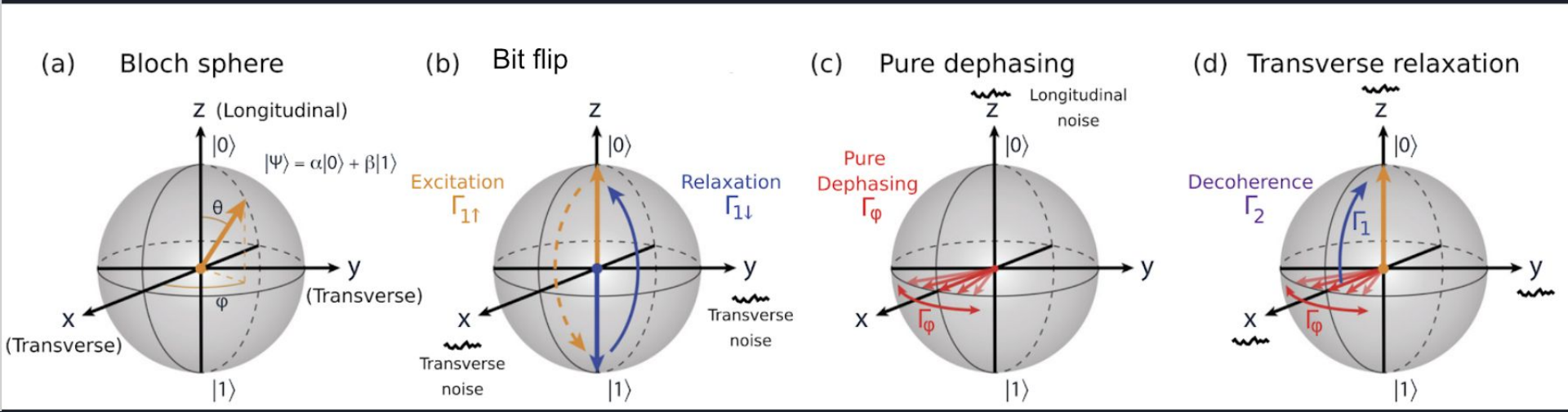
Note:

- Green vector is the initial state
- Black vector is the result after applying the black box.

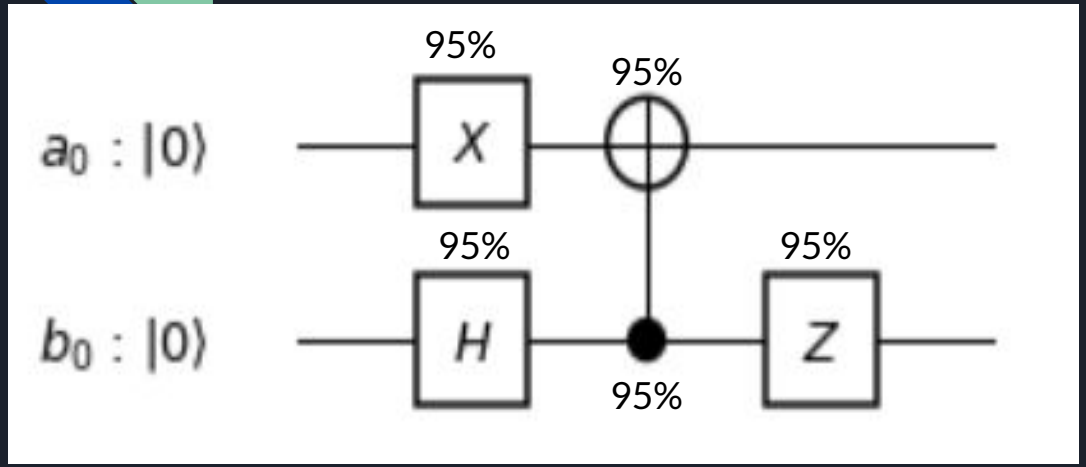
TRACK 4

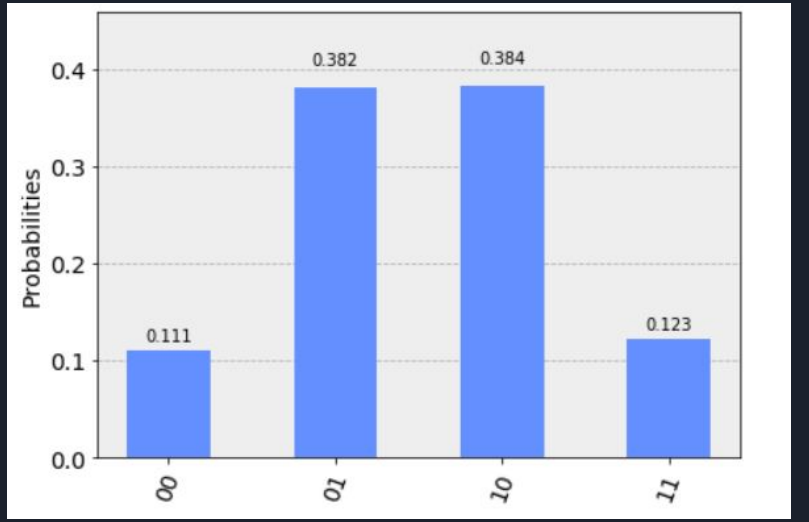
Jerry
Eric

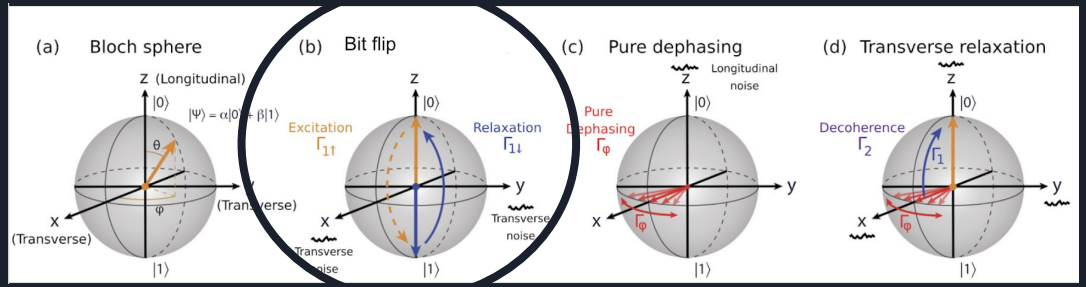
What is noise?



Bit-Flip - Singlet State (5% chance)

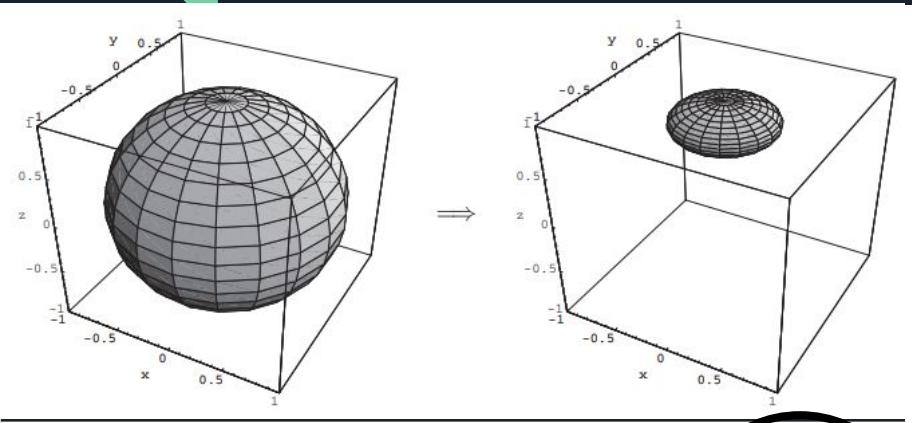


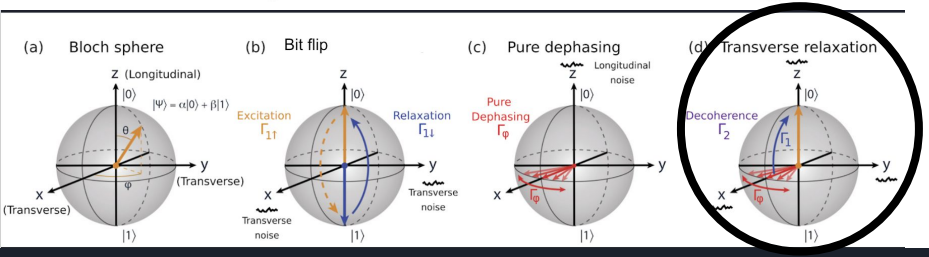


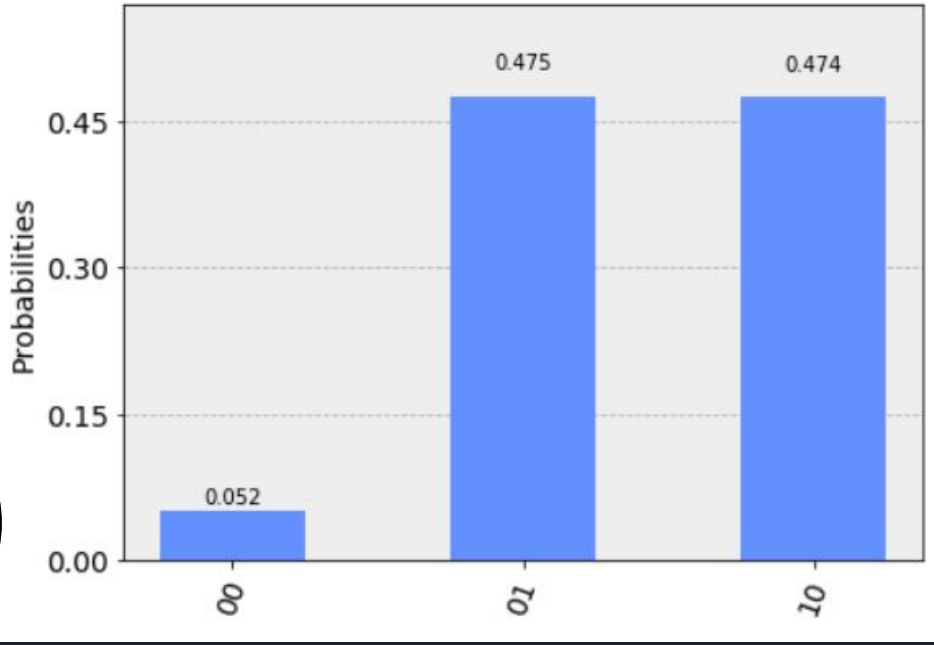


- Ideally, we expected a 50-50 probability distribution of 01 and 10.
- Erroneous results (00 and 11) have a total proportion greater than 0.05.
- Error accumulating with each gate.

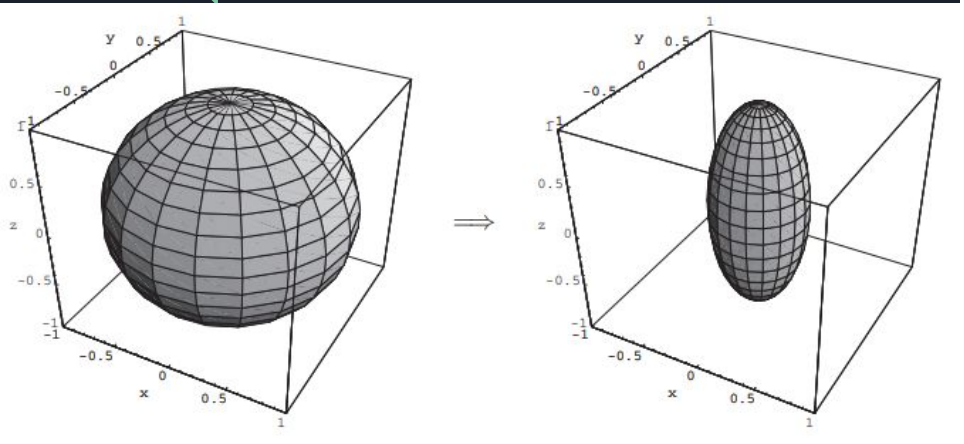
Transverse Relaxation

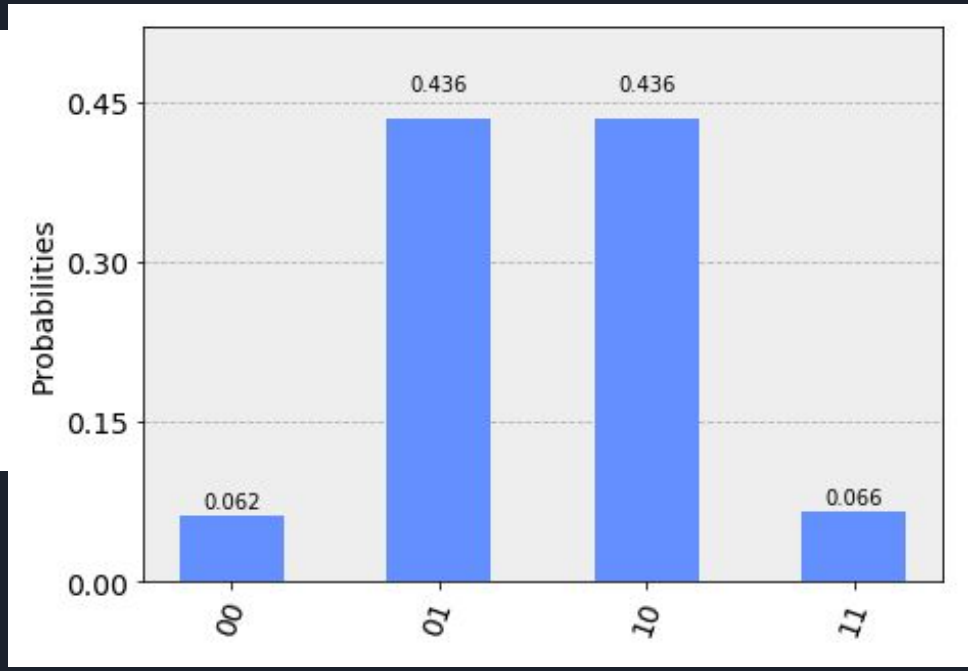






Phase-Flip (5% chance phase-flip)

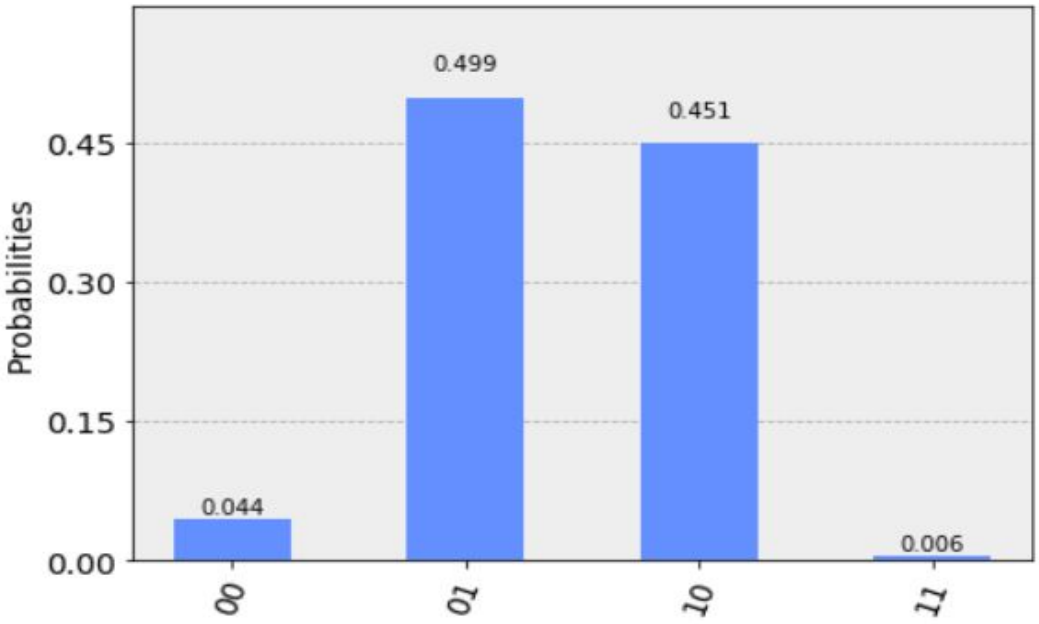




- We have rotated both qubits by 90 degrees along the y-axis to measure their x-components.
- Z-axis population errors were still present, but to a lesser extent.

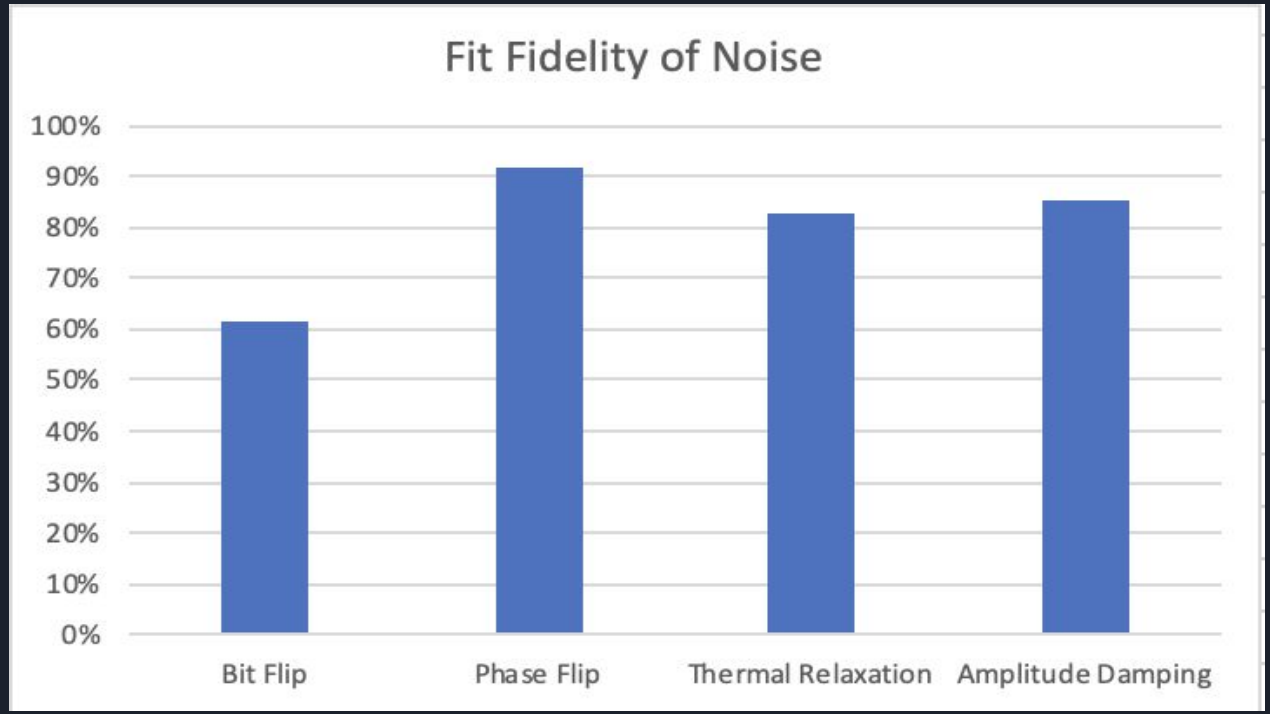
Thermal Relaxation

$$M = M_o (1 - e^{-t/T1})$$



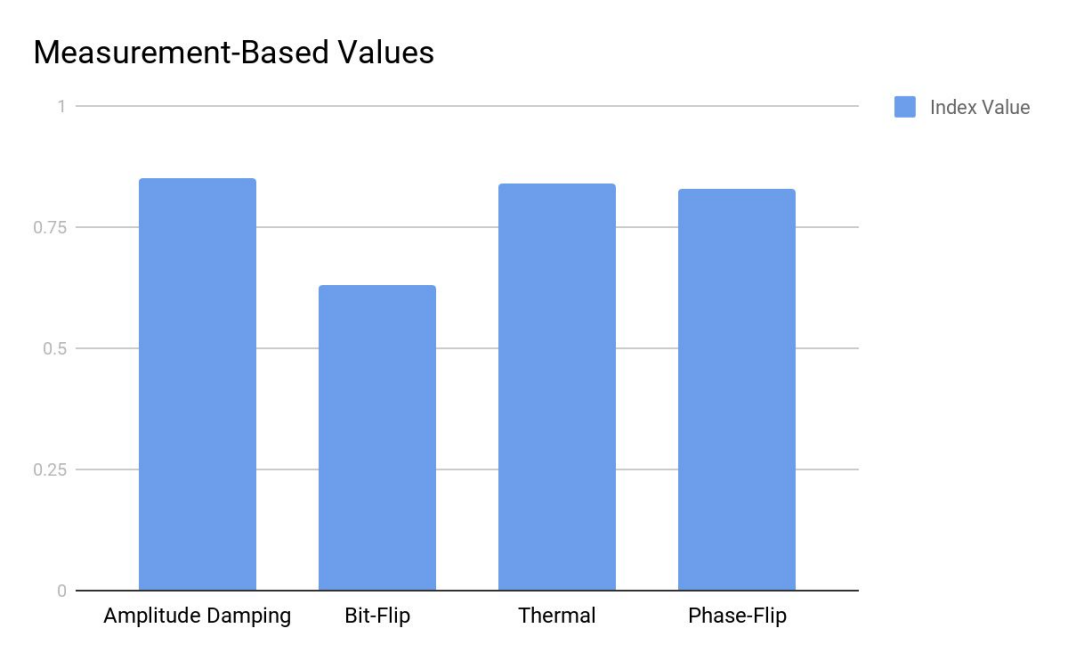
IBM QISKit Quantum State Tomography Fit Fidelity Method

- **Higher Fit Fidelities meant better results.**



Measurement-Based Error Comparison Method

$$1 - (P(00) + P(11) + |P(10) - P(01)| + P^*(11) + P^*(00))$$



Accounts for the difference between original 10/01 as well as phase effects.

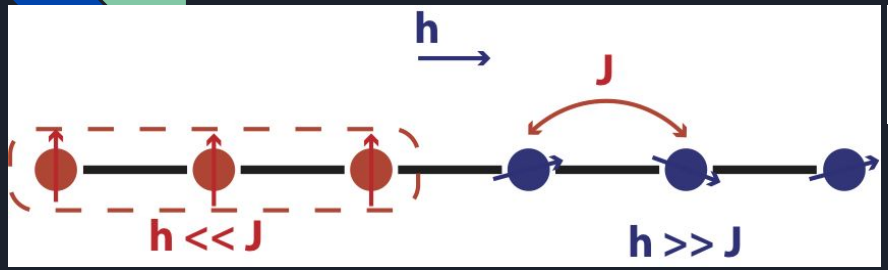
Higher indices closer to one are better.

Note that the last asterisked part means that a separate y-rotation is needed to measure x-components.

EMULATING MAJORANA CHAIN

Jerry and Devin

System Hamiltonian



$$H = -J\sum_i \sigma_{z,\,i}\,\sigma_{z,\,i+1} - \sum -ih_i\,\sigma_{x,\,i}$$

Z-axis interaction

Zeeman On-site interaction

Majorana are in **ferromagnetic** sites (low Zeeman field value of 0.01)

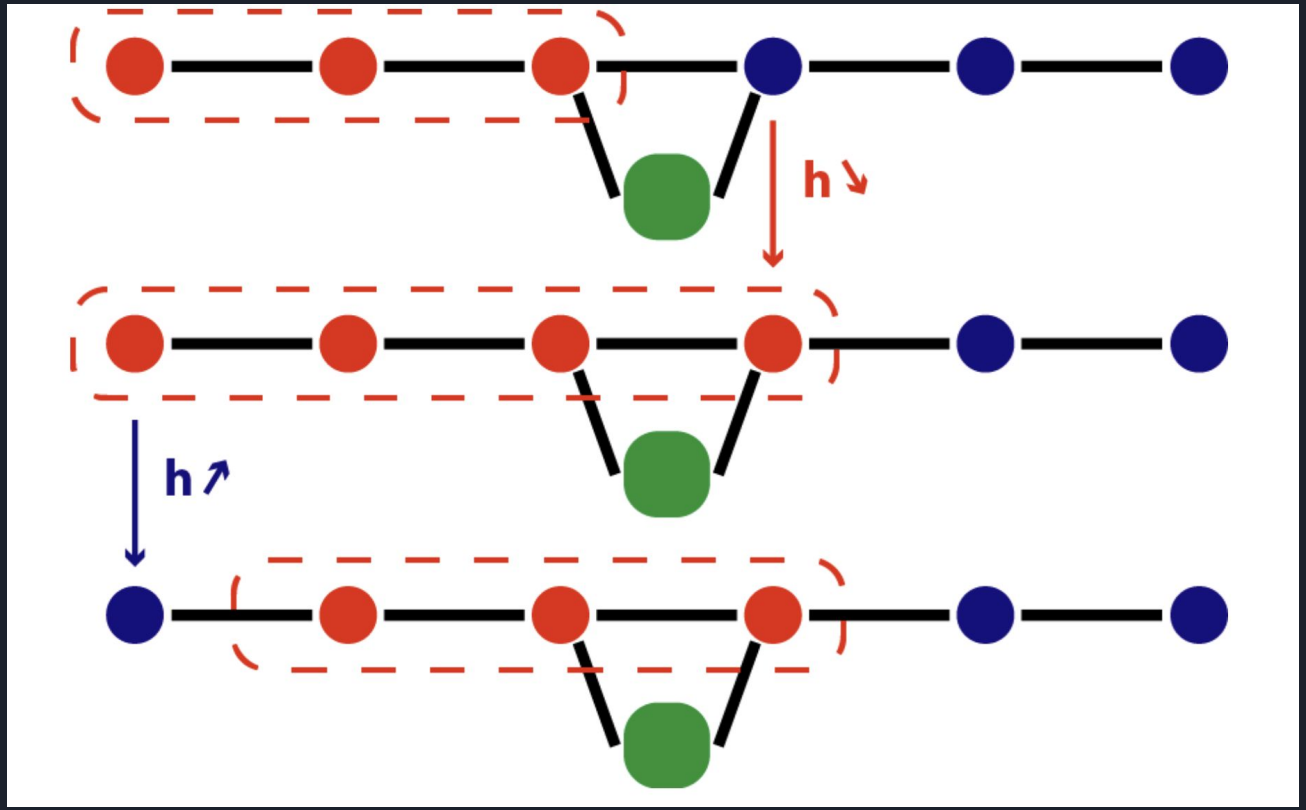
Paramagnetic sites are empty (high Zeeman field value of 10.00).

Two states:

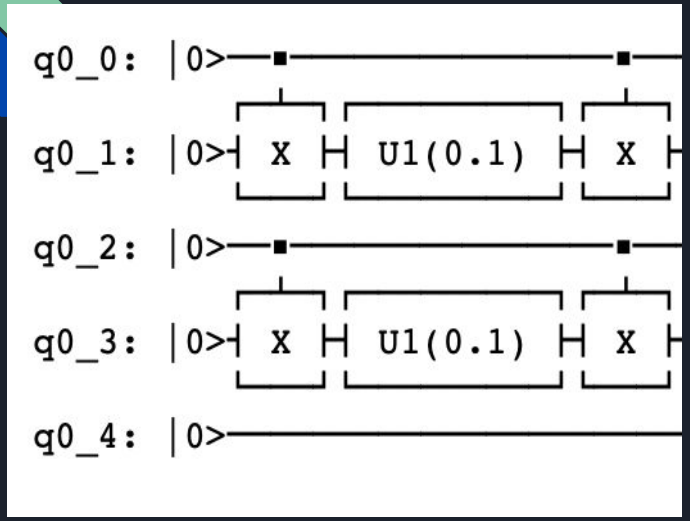
Logical zero |0> : (|↑↑↑> + |↓↓↓>)/√2 - measured as 000 after rotations

Logical one |1> : (|↑↑↑> - |↓↓↓>)/√2 - measured as 001 after rotations

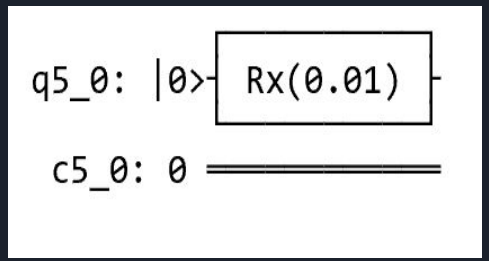
Moving Along the Chain



On-site and Interaction Hamiltonian




```python
def chain_hamiltonian(qc, q, coupling, zeeman, debug=False):
    """Implement the chain Hamiltonian.

    """
    n = len(zeeman)
    for j in range(0, n, 2):
        pair_interaction(qc, q, j, j+1, coupling)
    for j in range(1, n-1, 2):
        pair_interaction(qc, q, j, j+1, coupling)
        if(debug):
            qc.barrier()
    for j in range(n):
        zeeman_term(qc, q, j, zeeman[j])
    if(debug):
        qc.barrier()
```



Trotter Function

- The trotter function allows us to adiabatically change the quantum state:
  - |000> is the ground state, |111> is the excited state for the following code and examples

- Order is defined by:

$$e^{x(A+B)} = e^{p_1 x A} e^{p_2 x B} e^{p_3 x A} e^{p_4 x B} \cdots e^{p_M x B} + O(x^{m+1})$$

[2]

[2] Finding Exponential Product Formulas of Higher Orders

2nd Order Trotter

- The second order expansion is:

$$S_2(x) \equiv \mathrm{e}^{\frac{x}{2}A}\mathrm{e}^{xB}\mathrm{e}^{\frac{x}{2}A}$$ [2]

- Where A is the On-site Hamiltonian and B is the Interaction Hamiltonian

4th Order Trotter

- The Fourth order expansion is:

$$S_4(x) \equiv S_2(s_2x)^2 S_2((1-4s_2)x) S_2(s_2x)^2$$ [2]

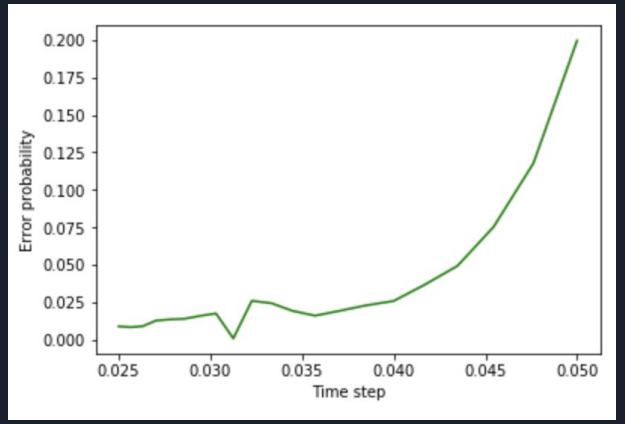
Circuit Depth

- Circuit depth is how many necessary gates an operation takes up

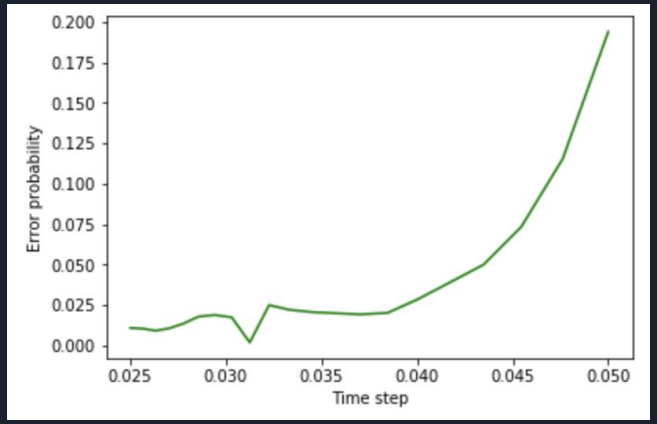
- By implementing higher order Trotter expansions we can lower the circuit depth while maintaining or increasing accuracy

- We aim to lower the depth within the range of the number of functioning qubits on a real quantum computer

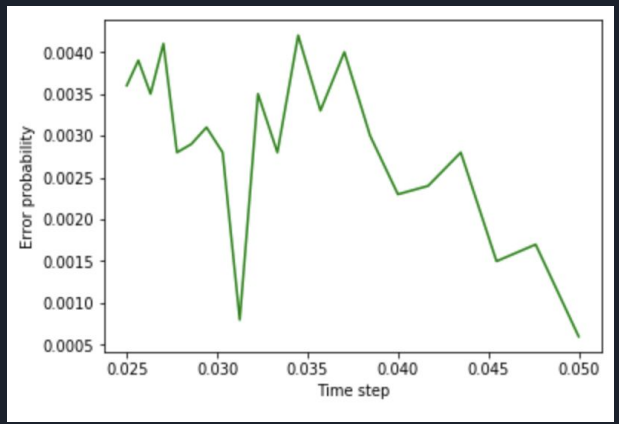
Results GSE - Error vs. Time Step (Keeping total time constant)



1st order

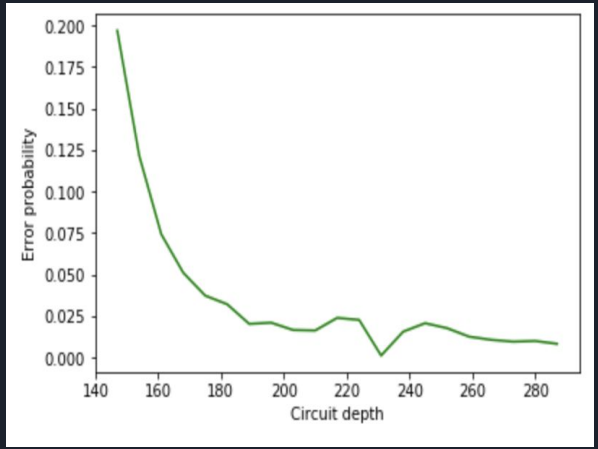


2nd order

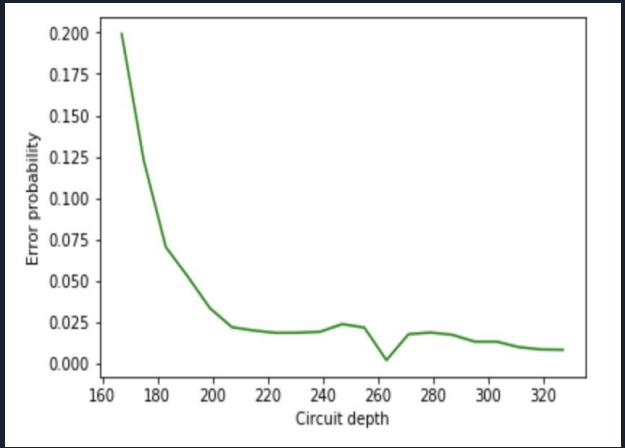


4th order

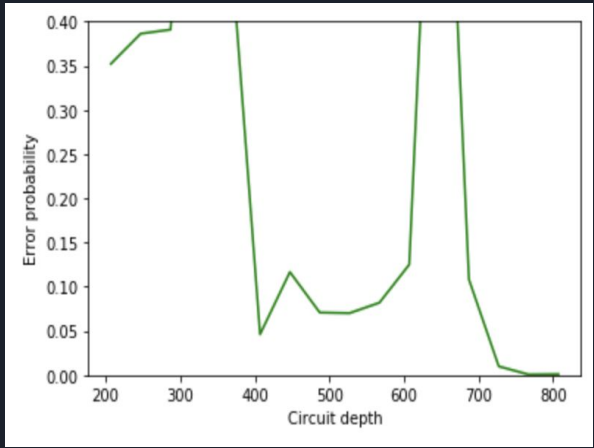
Results GSE - Error vs. Circuit Depth



1st order



2nd order



4th order

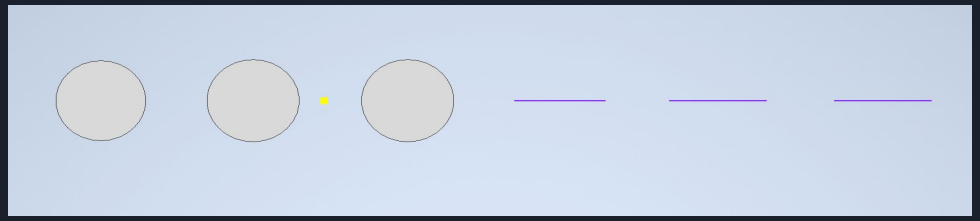
Move Chain

Top: Move the chain from the first three sites to the last three sites (6 qubits total required). Requires a change in ferromagnetic and paramagnetic sites.

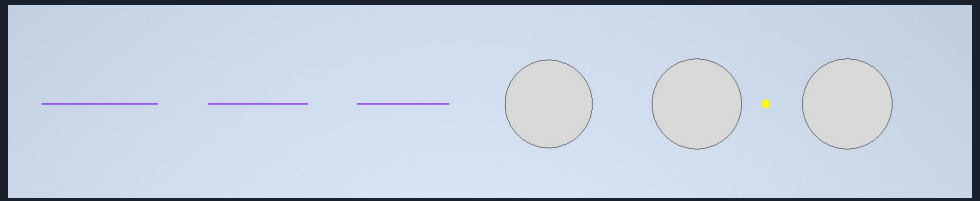
Bottom: Wave Diagram

Gap fraction is the fraction of the gap to move along the H-field by. Higher gap fractions lead to faster changes.

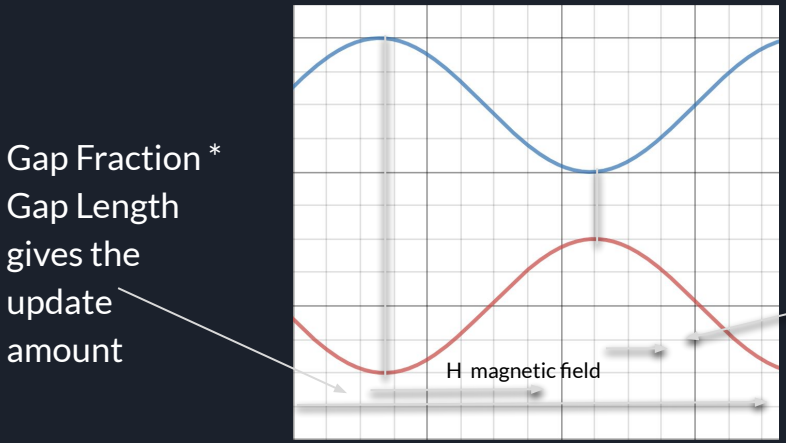
Minimal increment is the minimal increment in case the gap is too small for a significant change or zero.



Initial State



Final State



Testing Time Durations on Moving the Majorana Chain

```
initialize_chain(qcirc, qreg, initial_config, 'logical_zero')
qcirc.draw()
move_chain(qcirc, qreg, initial_config, final_config, 0, 0.25, 0.25, 2, 10, method = "both")
qcirc.depth()
```

```
def move_chain(circuit, qreg, initial_zeeman, final_zeeman, coupler_inter,
               gap_fraction, min_increment, delay, trotter_step_number,
               method='both'):
```

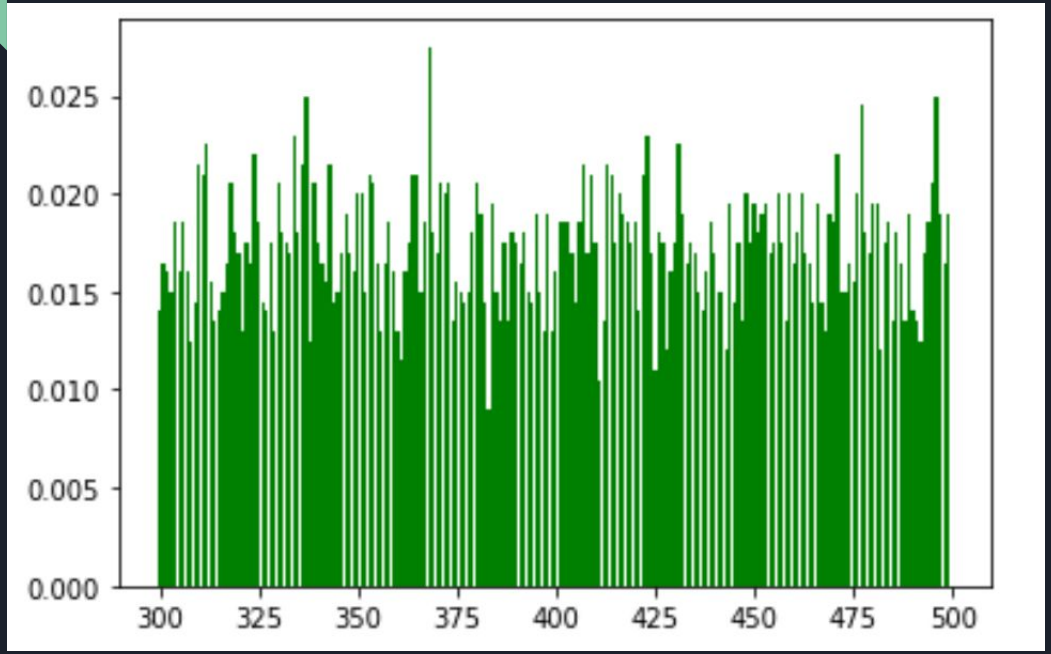
Test the efficacy of various different gap fractions and minimum increments by changing one of gap_fraction and min_increment and holding the other one fixed at 0.25.

We expected to discover a close-to-1 proportion of 000 (which represents the logical-zero state) when we measure the last three qubits

Error Rate Overall for Varying Gap Fraction

Error Value

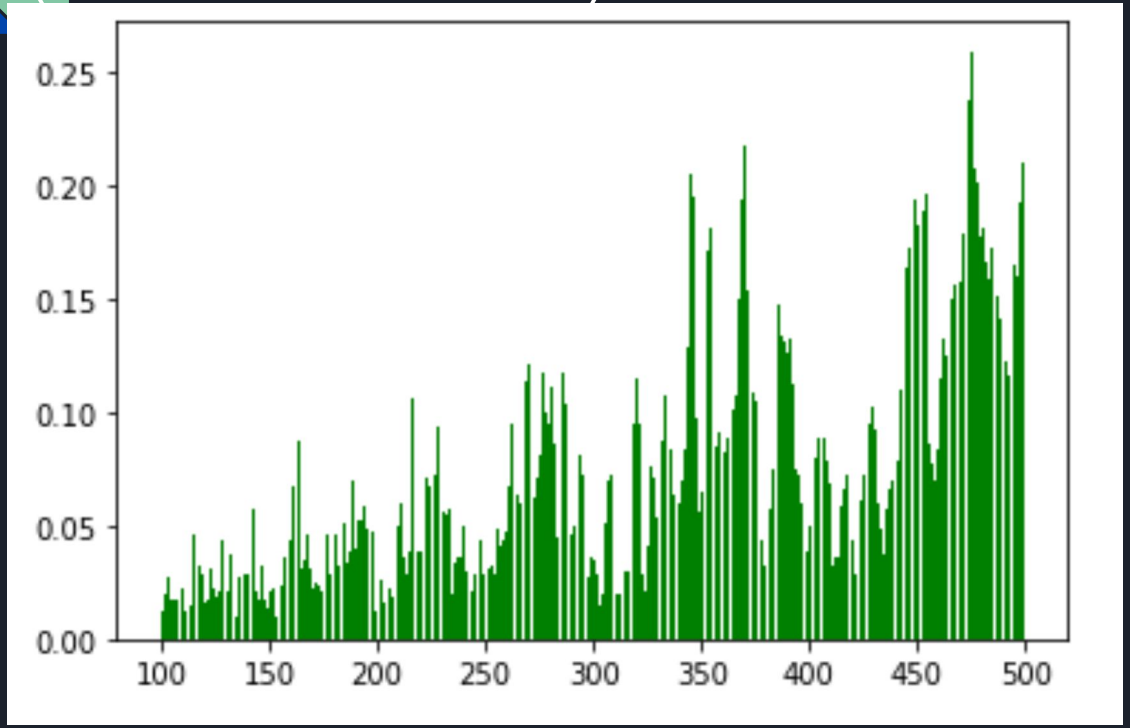
(Not 000)



1000x Gap Fraction Value

Given that the Min_Increment was fixed, the error rate stayed at a relatively low level for all values of gap fraction - no matter how large the gap fraction was (ranging from 0 to 1).

Error Rate Overall for Varying Min Increment (Value of 0.1 to 0.5)



Error Rate

(Not 000)

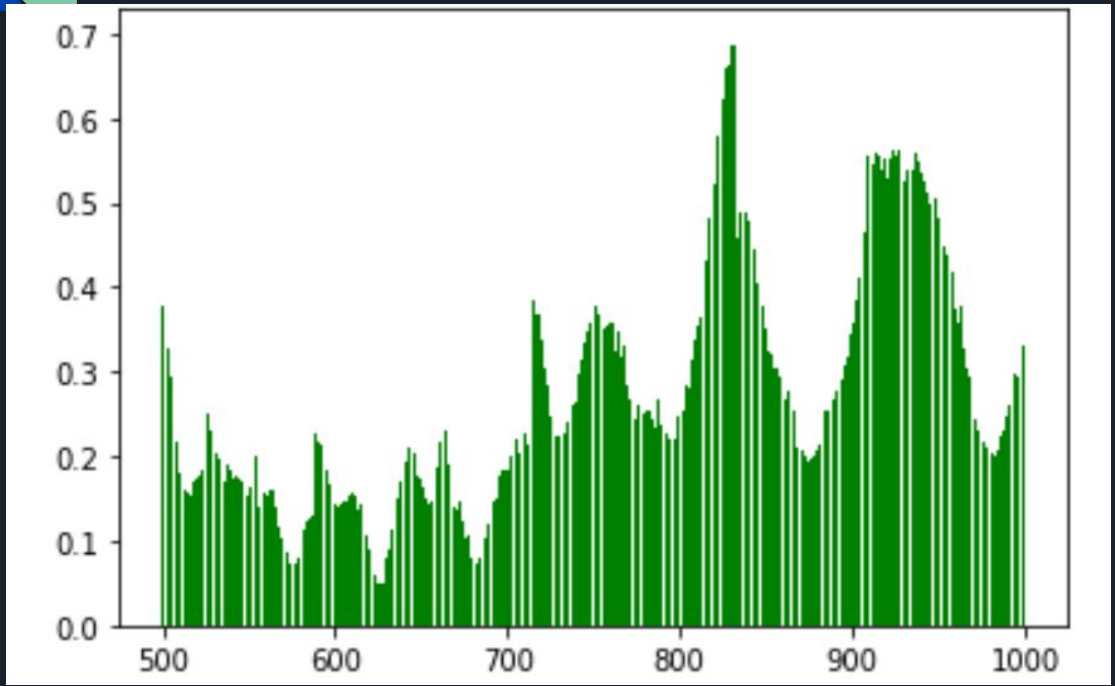
1000x Minimum Increment Value

In order to determine the ideal combination, we attempted to use the highest possible value of a minimum-increment that still provided a high fidelity.

Error Rate for Min Increment (Continued with values 0.5 to 1)

Error Value

(Not 000)



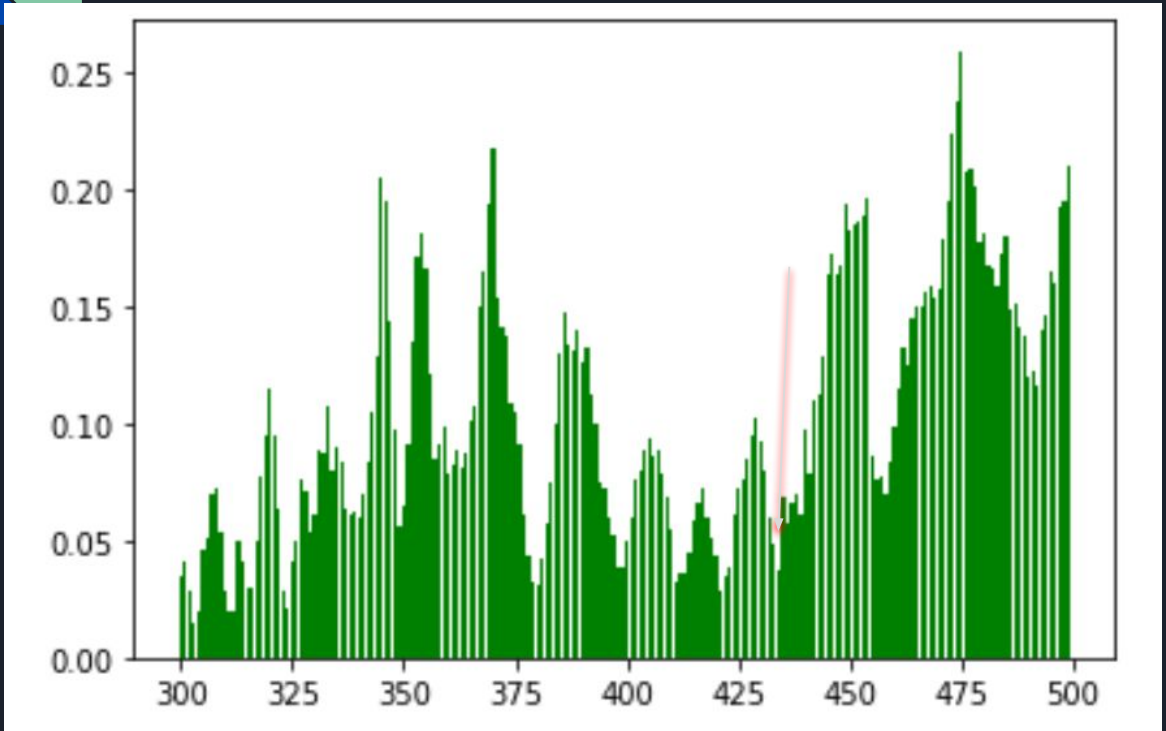
1000x Minimum Increment Value

To the left are histogram results for higher values of minimum increment. Error rates started to consistently exceed 10% after around 0.7.

Close Up of 0.3 to 0.5

Error Rate

(Not 000)



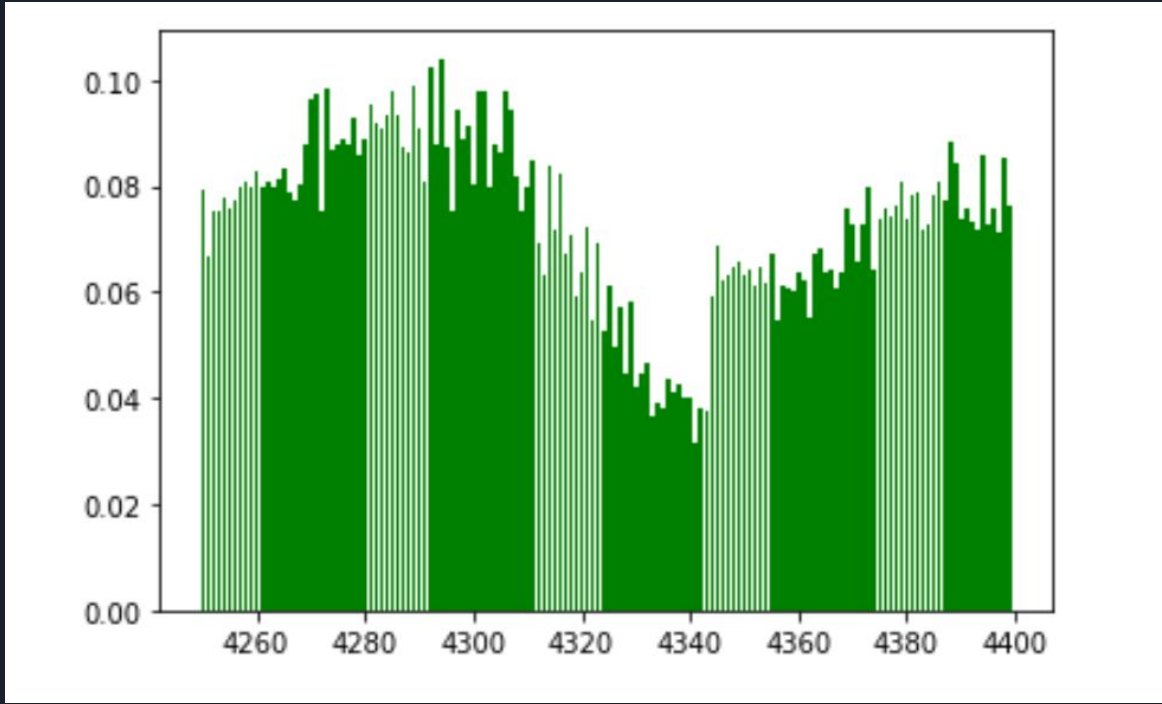
1000x Minimum Increment Value

The arrow points to the maximal minimum increment that we can use and still have an error rate that is significantly below 0.05.

Further Close Up of Ideal Zone: 0.425 to 0.44

Error Rate

(Not 000)



10000x minimum increment value

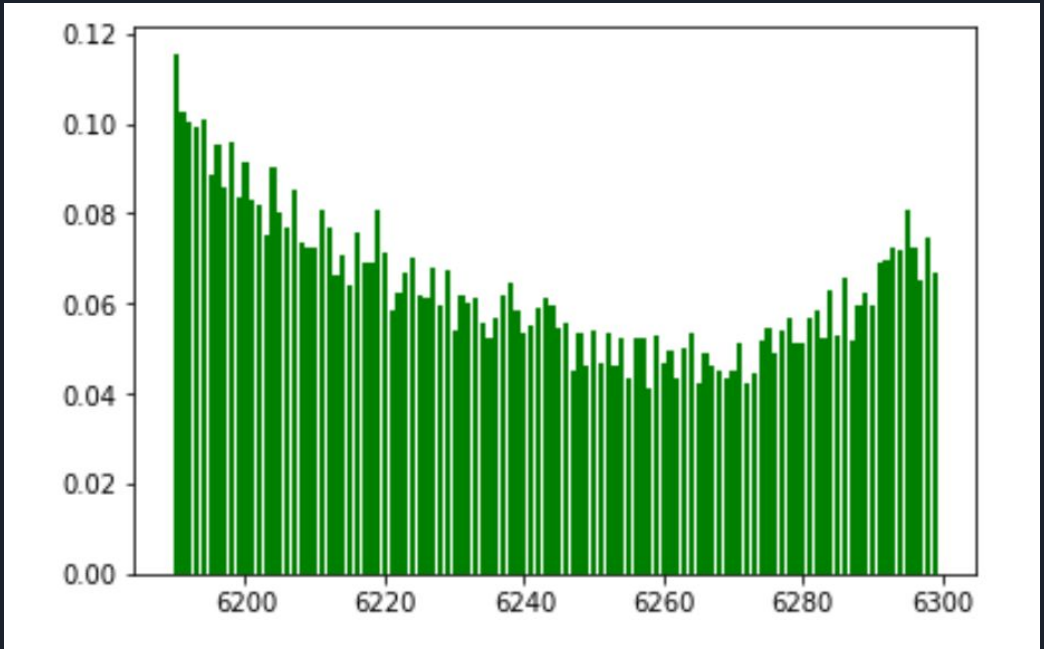
Using a very precise increment of 0.0001 and then measuring the error rate from 0.425 and 0.439, the lowest error rate occurred at 0.434, an error rate of around 3%.

This error rate was relatively constant for **all values of gap fraction, when keeping the min_increment at 0.434.**

Further Close Up of Final Ideal Zone: 0.619 to 0.63

Error Rate

(Not 000)



10000x minimum increment value

Ideal zone occurs at a minimum increment of approximately 0.625. Error rate is slightly higher than the second-to-last ideal zone, slightly less than 0.05.

Conclusions

- For simpler operations going to higher order expansions is unnecessary due to the increased circuit depth and marginal increase of accuracy
- For more complex operations higher order trotters could be used better due to the wider possible range in circuit depth
- Even though the circuit depth can be reduced it is still too high to work on a real quantum computer for operations like move chain

- Move chain fidelity ONLY depended on min_increment. **Fastest and lowest error min_increment values were exactly 0.434 and 0.626.**

HARDWARE

THREE AXIS HELMHOLTZ COIL BLOCH SPHERE

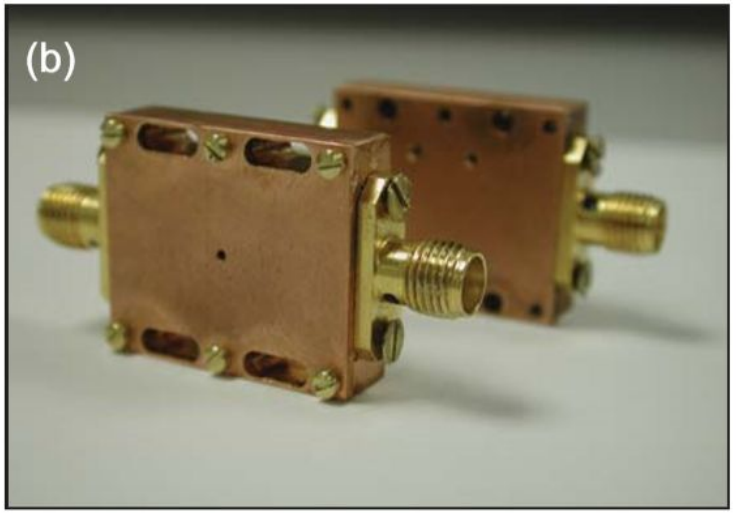
MECHANICAL BLOCH SPHERE

SAPPHIRE BOX

Jerry

Introduction to the Sapphire Box:

- The box itself is made of copper, with two highly conductive sapphire substrates inside
- The box acts as a heat sink due to the ability of the copper to absorb heat. Absorbing excess heat that is released reduces the noise in experiments.
- SMA connectors on both sides attached to copper wires in a coaxial line, allowing the interior to act as a heat sink.

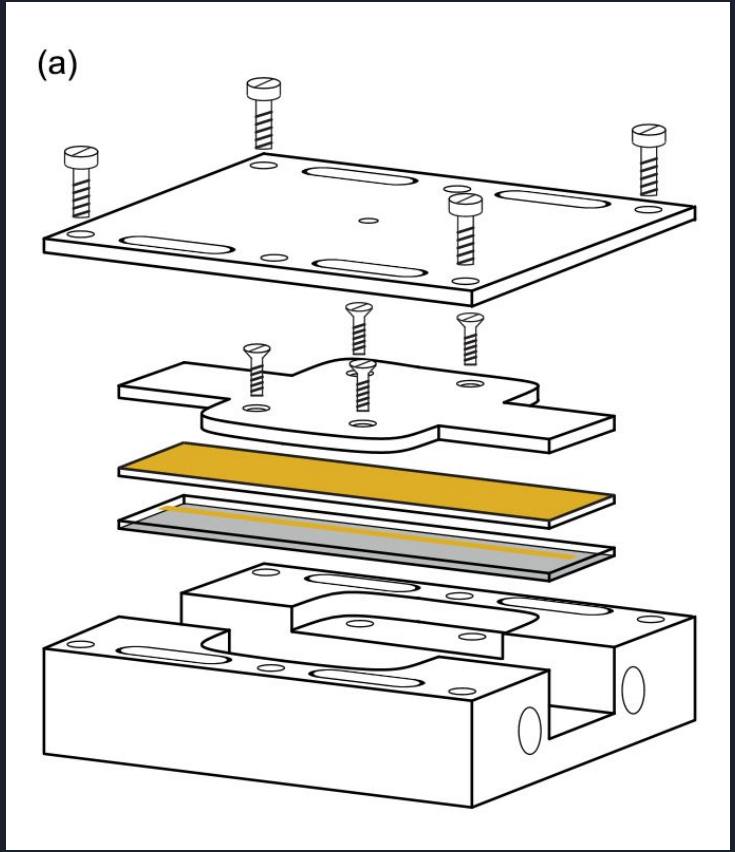


Methodology:

- Used the model in Leonardo Di Carlo's Harvard PhD Thesis. The article **only provided the dimensions of the two substrates (thin colored boards to the right).**

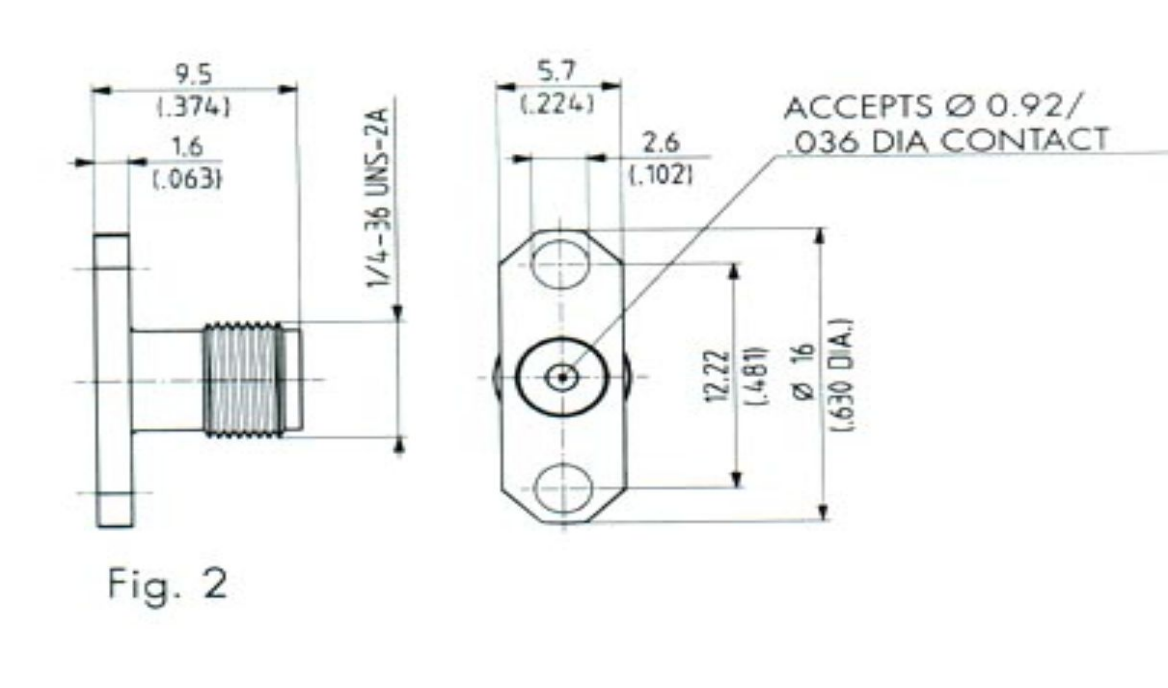
- Measured the ratio of the length of the other parts to the length of the substrate to get the other lengths.

- Used Autodesk Inventor (AUTOCAD) to create 3D models.

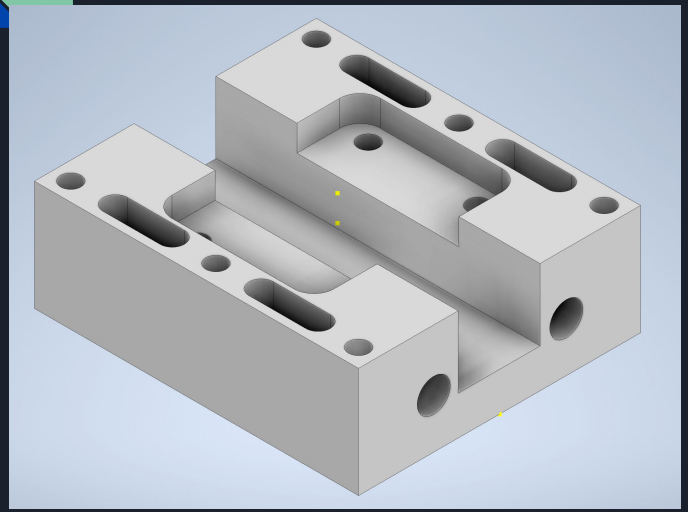


SMA Connector Dimensions

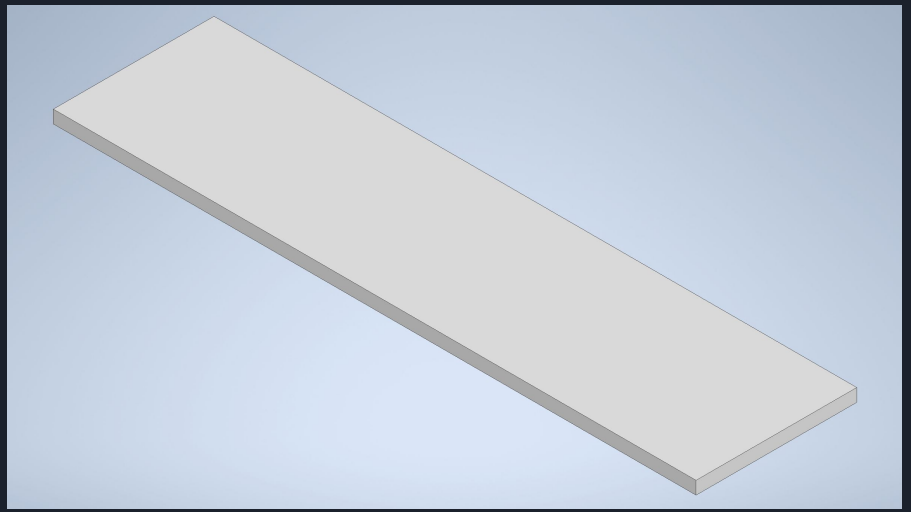
- Holes on the side of the box are larger than the regular holes. I searched up a 3D model of the SMA Connector (23-SMA-50-0-166) online and found the length of the diameter of the hole (0.102 inches).



The Parts in AutoDesk

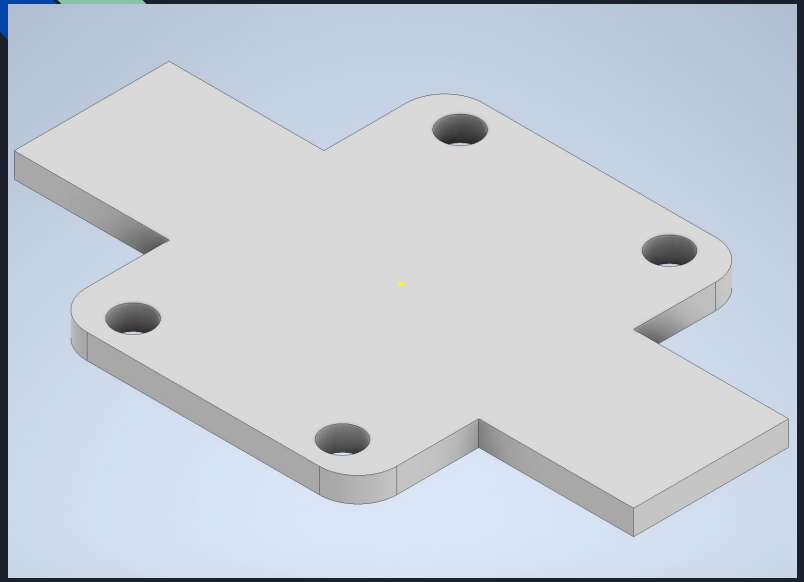


Bottom Box

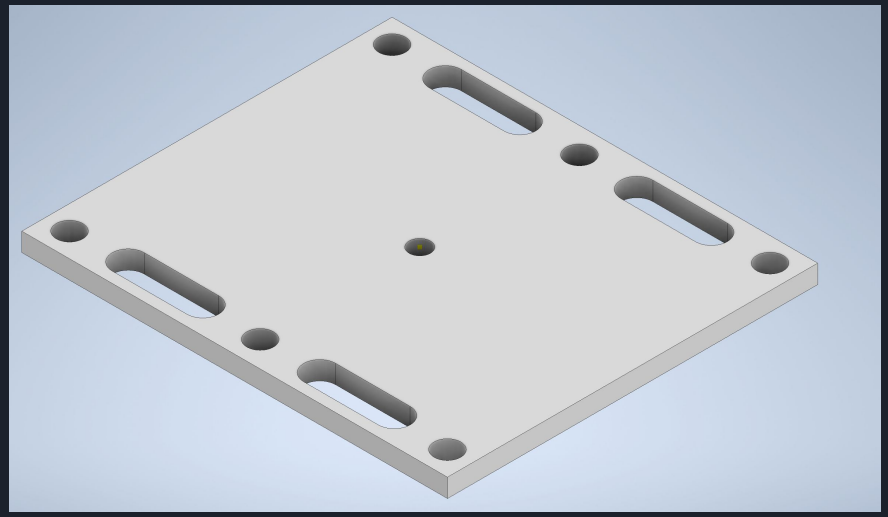


Sapphire Substrate Board

The Parts in Autodesk



Middle Section

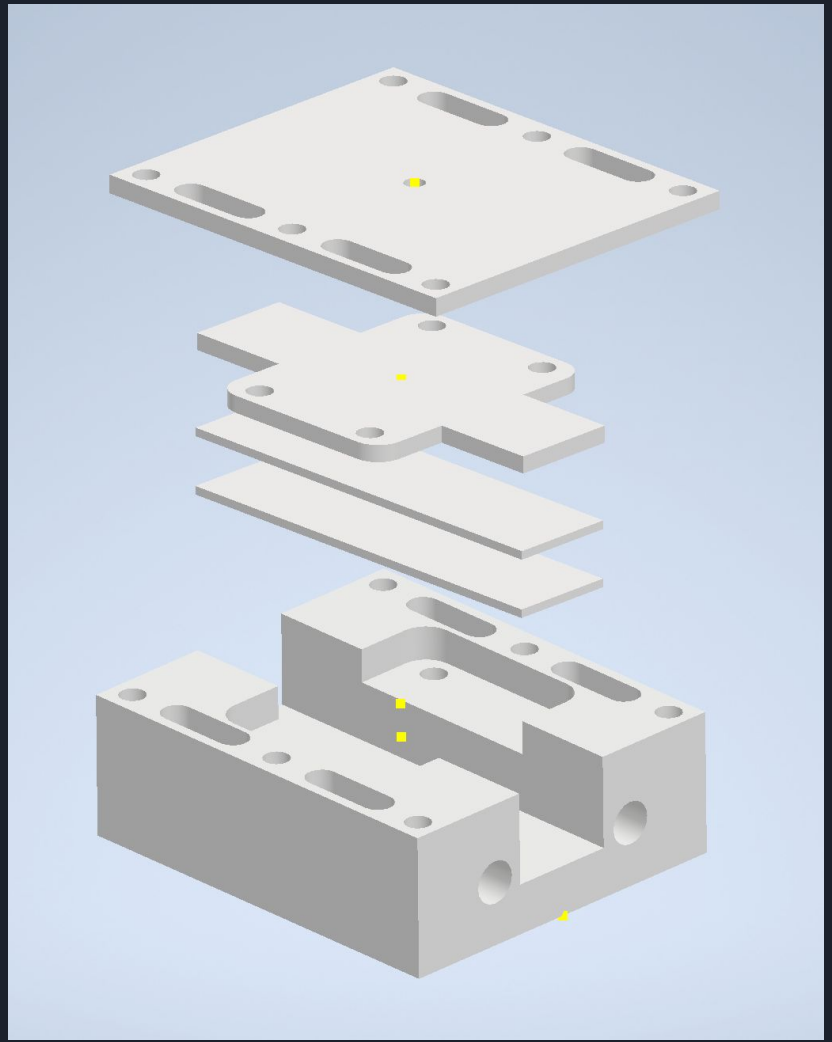


Top Section

Exploded View

- The parts are screwed together through the circular holes by screws.

- The presence of many holes allows for increased air-flow in the box and greater conductivity.

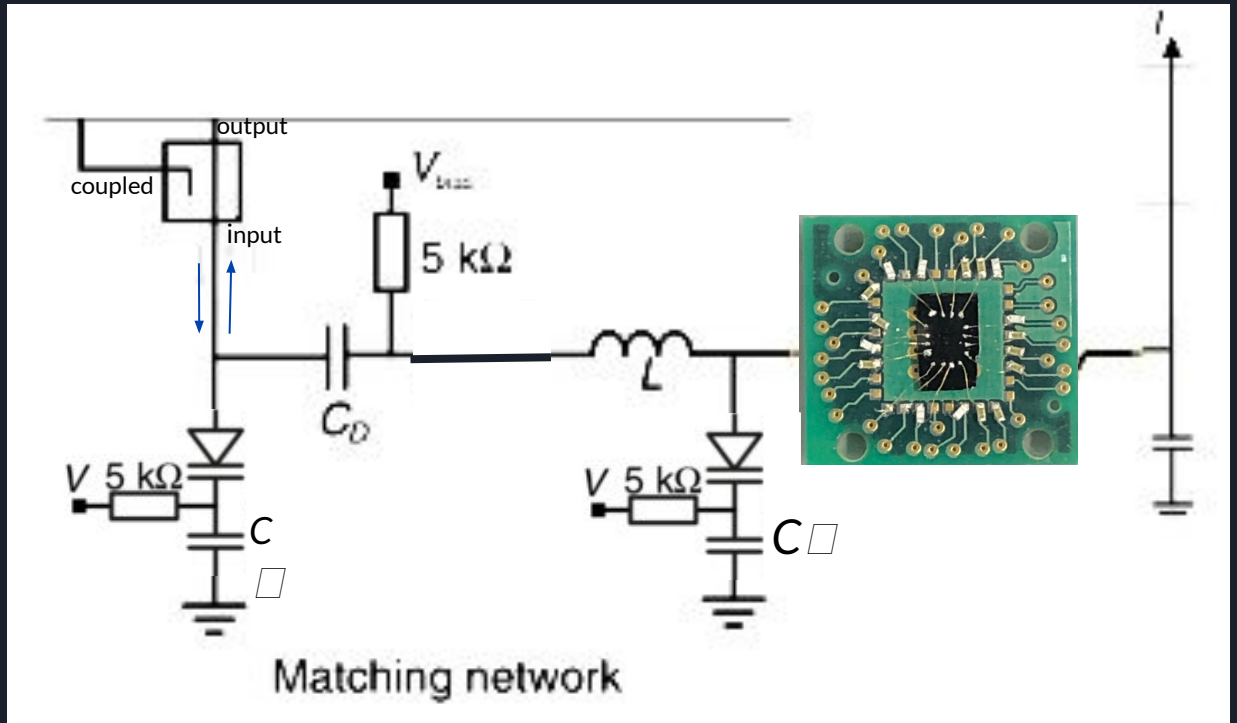


RESONANT CIRCUIT

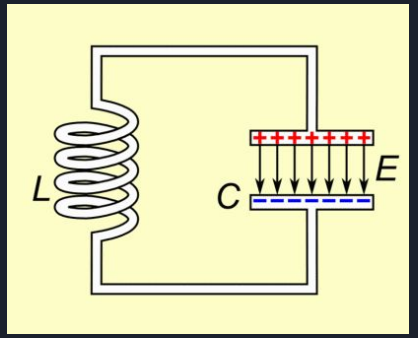
Eva
Abigail
Maheshi

Goal:

Design, manufacture and test an impedance matching resonant circuit using voltage-tunable varactors that cancel out parasitic capacitances.



Resonant Circuit

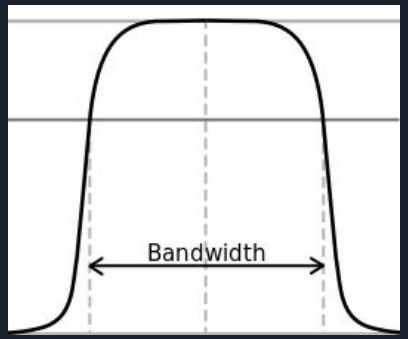


Resonance allows for the maximum power output of an RLC circuit at the frequency:

$$f_r = \frac{1}{2\pi\sqrt{LC}}\,(\text{Hz}) \quad \text{or} \quad \omega_r = \frac{1}{\sqrt{LC}}\,(\text{rads})$$

The quality factor is a parameter to describe how sensitive a resonator is.

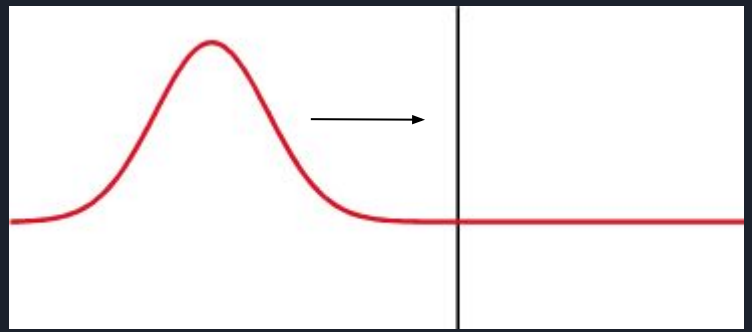
$$Q \stackrel{\text{def}}{=} \frac{f_r}{\Delta f} = \frac{\omega_r}{\Delta\omega}$$

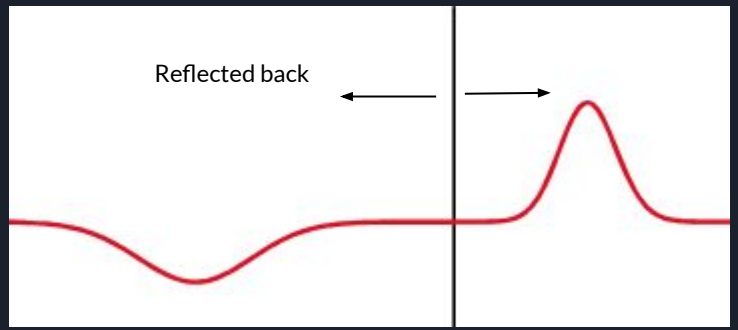


Impedance Matching

Design the input impedance of an electrical load or the output impedance of its corresponding signal source so that it maximizes power transfer or minimizes signal reflection.

$$\Gamma_{12} = \frac{Z_2 - Z_1}{Z_2 + Z_1}$$



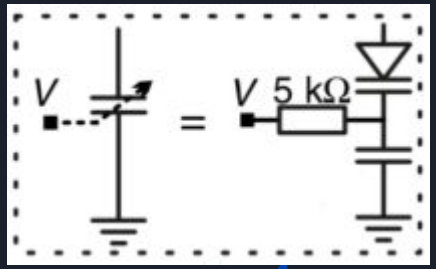


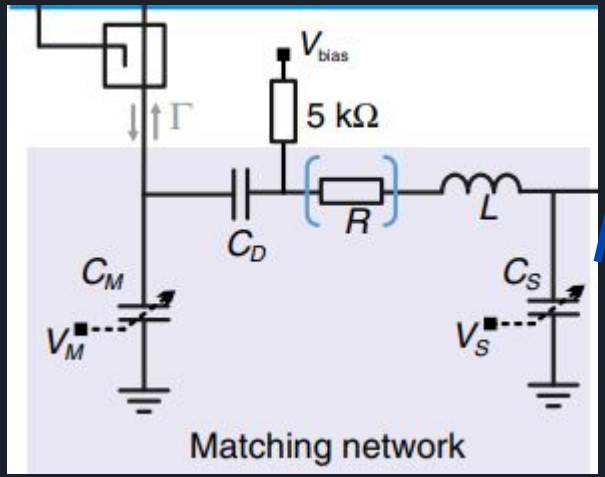
Variable Capacitors

We can vary capacitances using varactors.

Varactors diodes are semiconductor microwave solid-state devices that allow variable capacitance through voltage control.

Varactors can be explained as P-N junctions.





Closer look at Varactors

Capacitance is directly proportional to the area of the plates.

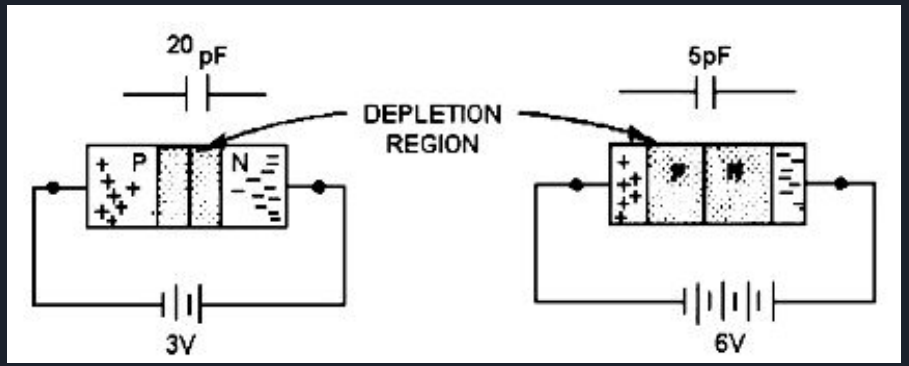
When reverse bias is applied, the depletion area between the P-type and N-type regions can be considered as the dielectric.

The volume of the depletion region varies with the change in reverse bias.

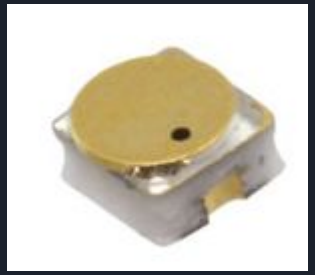
$$C = \frac{\varepsilon A}{d}$$

$$C_J = \kappa\varepsilon_0 \frac{A}{w(v_R)}$$

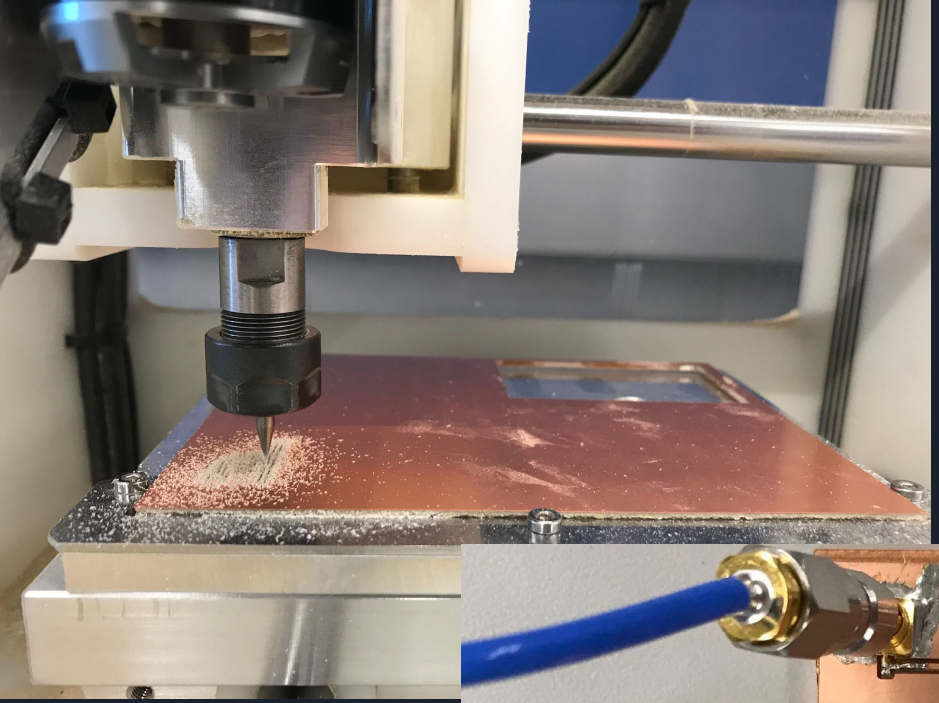
A- Area
d - distance between plates
w- depletion width
Vr- reverse bias

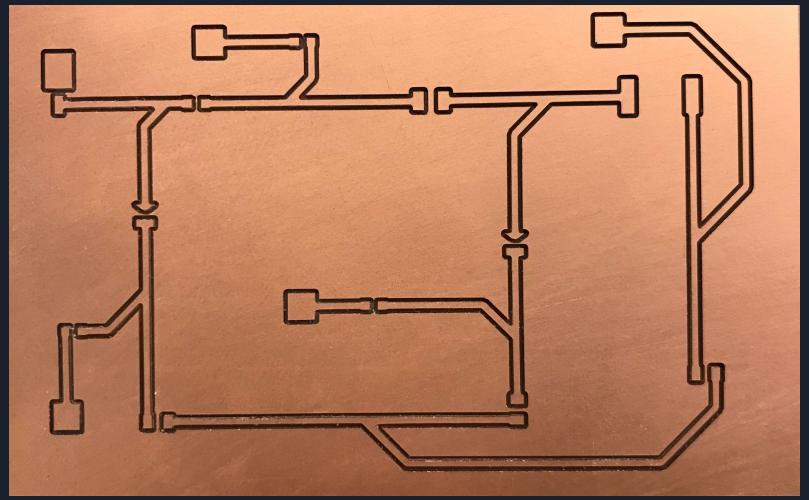


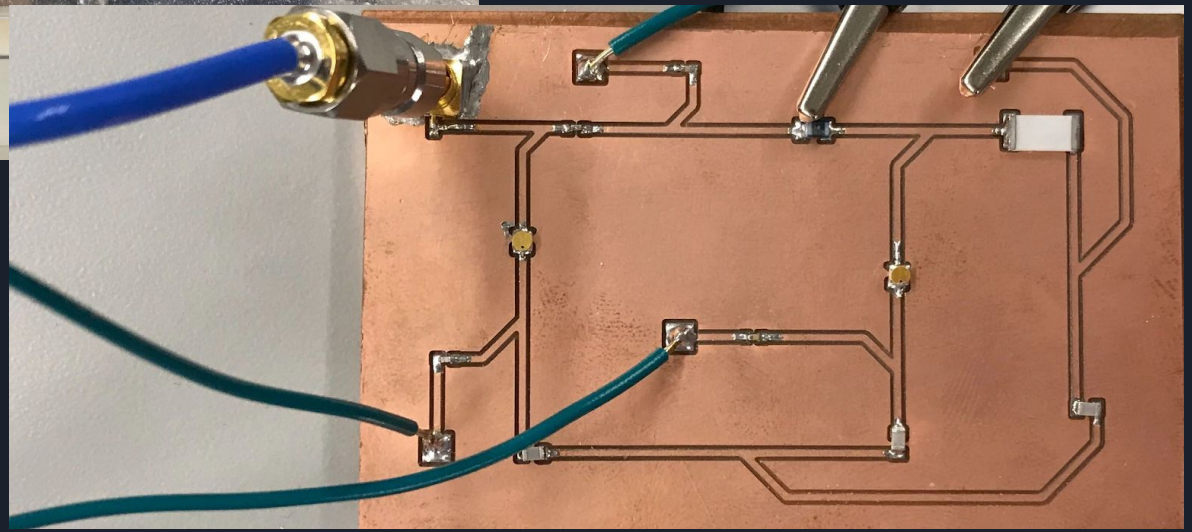
Materials



- Varactor
- Capacitor 100pF ,82pF
- Resistor 1G ohm, 5kohm
- Inductor
- Directional Coupler

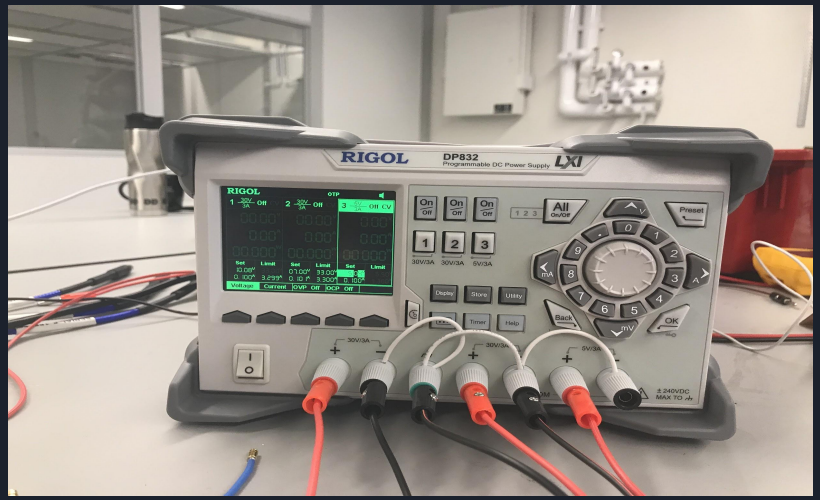


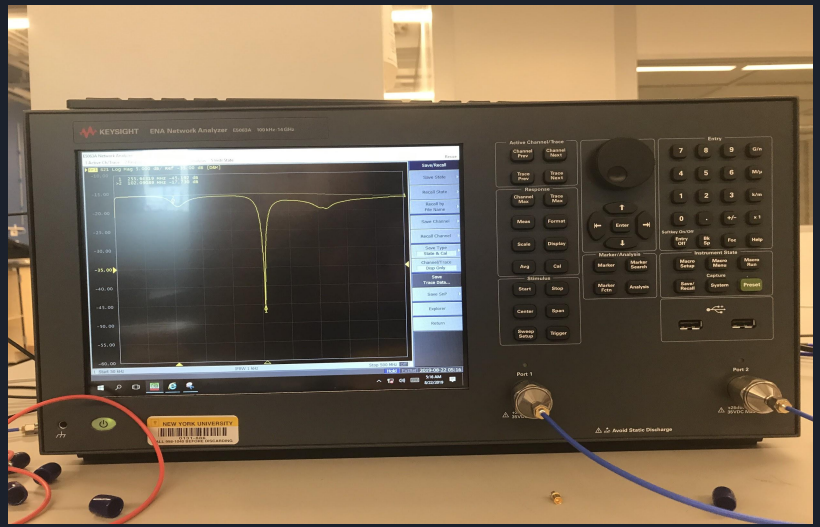


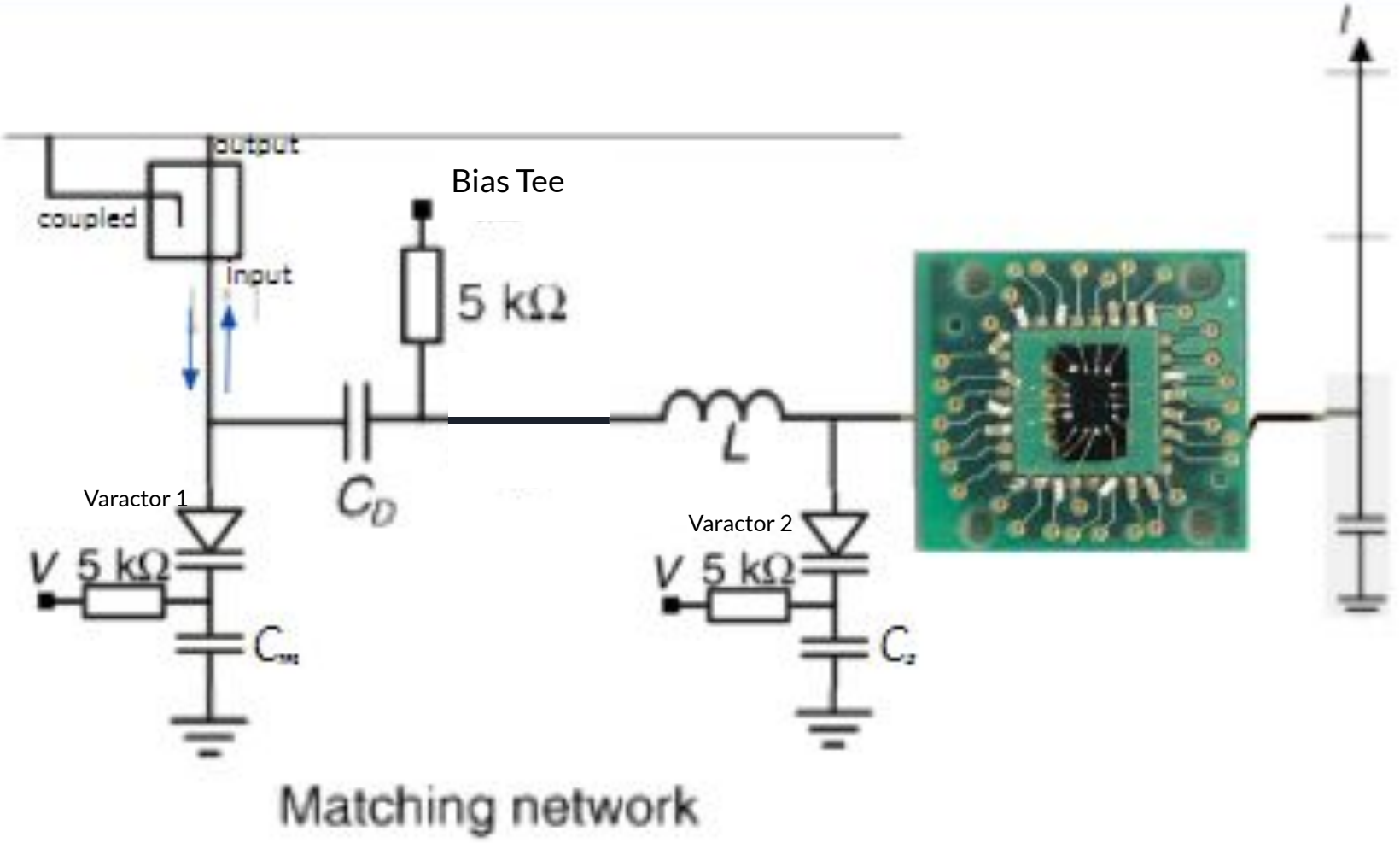


Data Collection

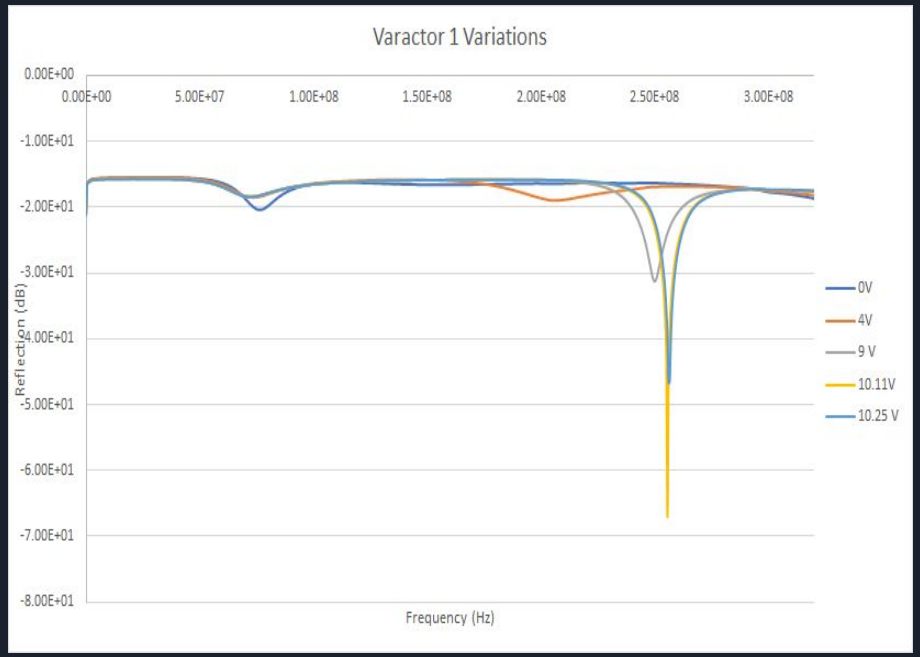
- Set up Vector Network Analyzer
  - VNA provides a stimulus for the network and then monitors the response
  - Measures amplitude response and phase



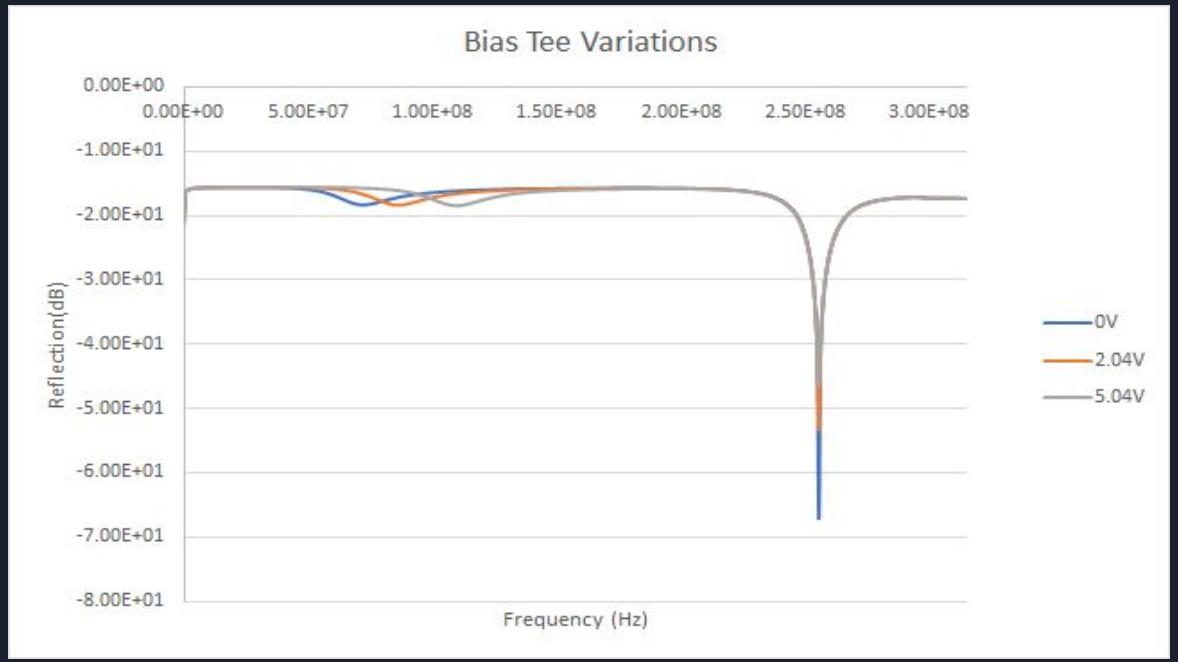



output
coupled
input
Bias Tee
5 kΩ
C_D
L
Varactor 1
V 5 kΩ
Varactor 2
V 5 kΩ
I
Matching network

Effect of Varactor Bias on the Circuit Resonance



Effect of Bias Tee Voltage on the Circuit Resonance



Varactor 1 bias: 10.11 V

Varactor 2 bias: 0 V

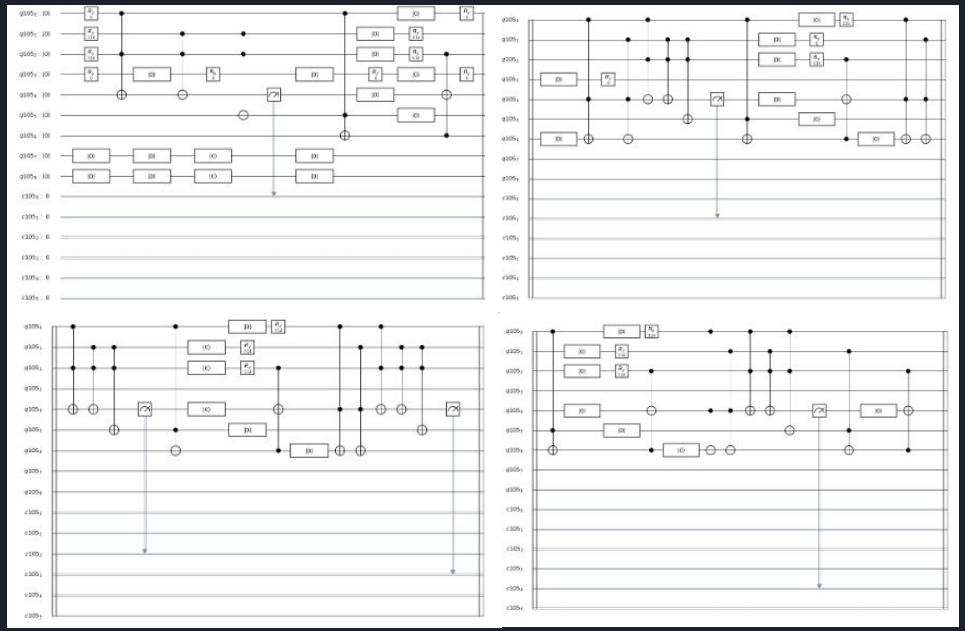
THANK YOU FOR YOUR ATTENTION!

CALCULATOR

A demonstration of Quantum Error Detection And Correction

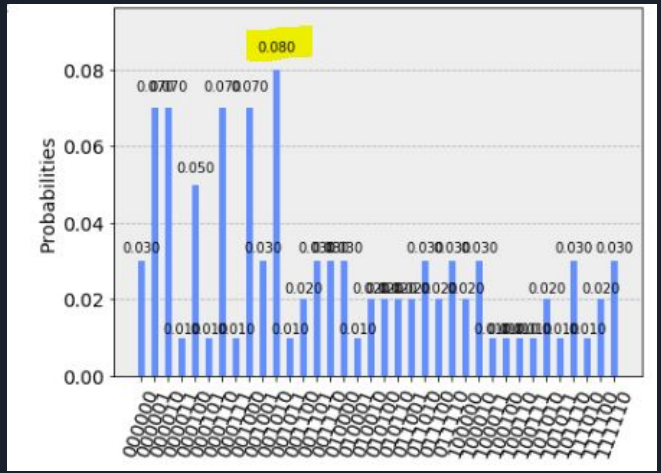
My Addition Circuit

When I had created a circuit with qiskit that could perform addition , it ended up looking like this



Adding Noise

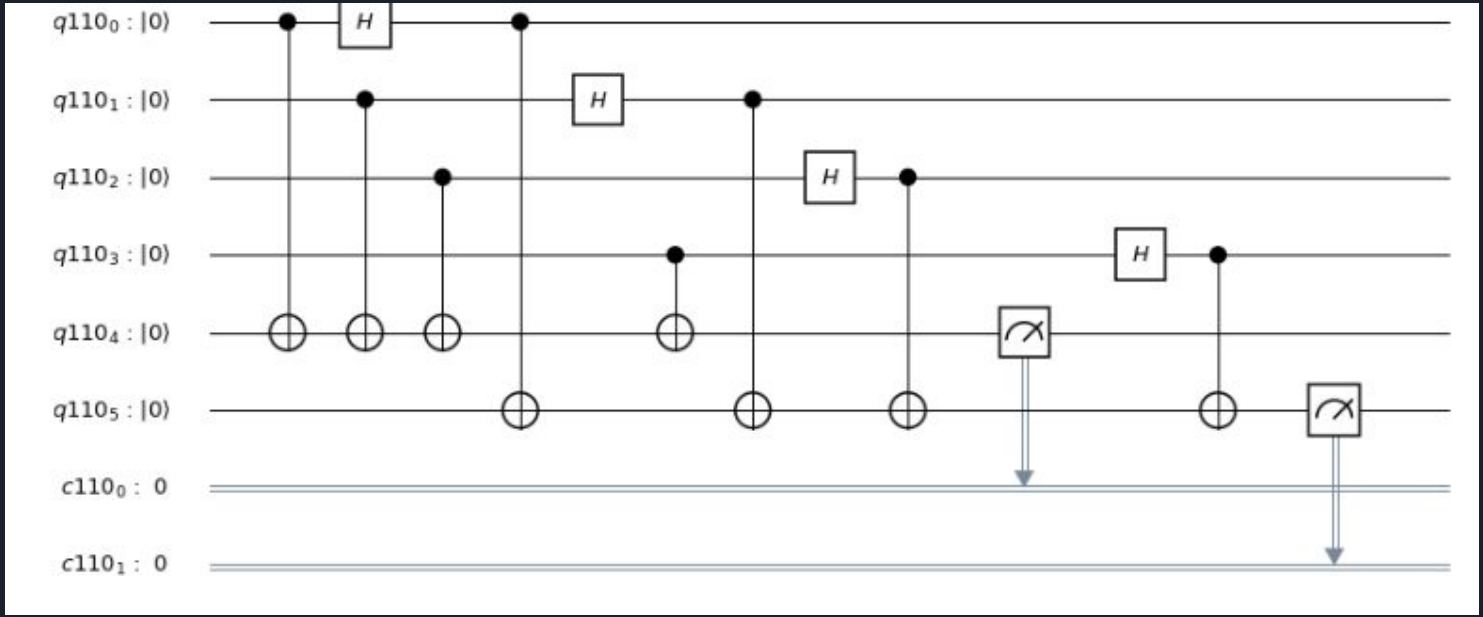
Although this calculator worked correctly for all numbers that it could calculate, it was not an accurate representation of a real quantum system. Real quantum systems have noise and errors, this is an ideal representation. So what would my results look like with some noise in the system. I created a noise model with reset, gate, and measurement error probabilities of 1%, with an expected answer of ten



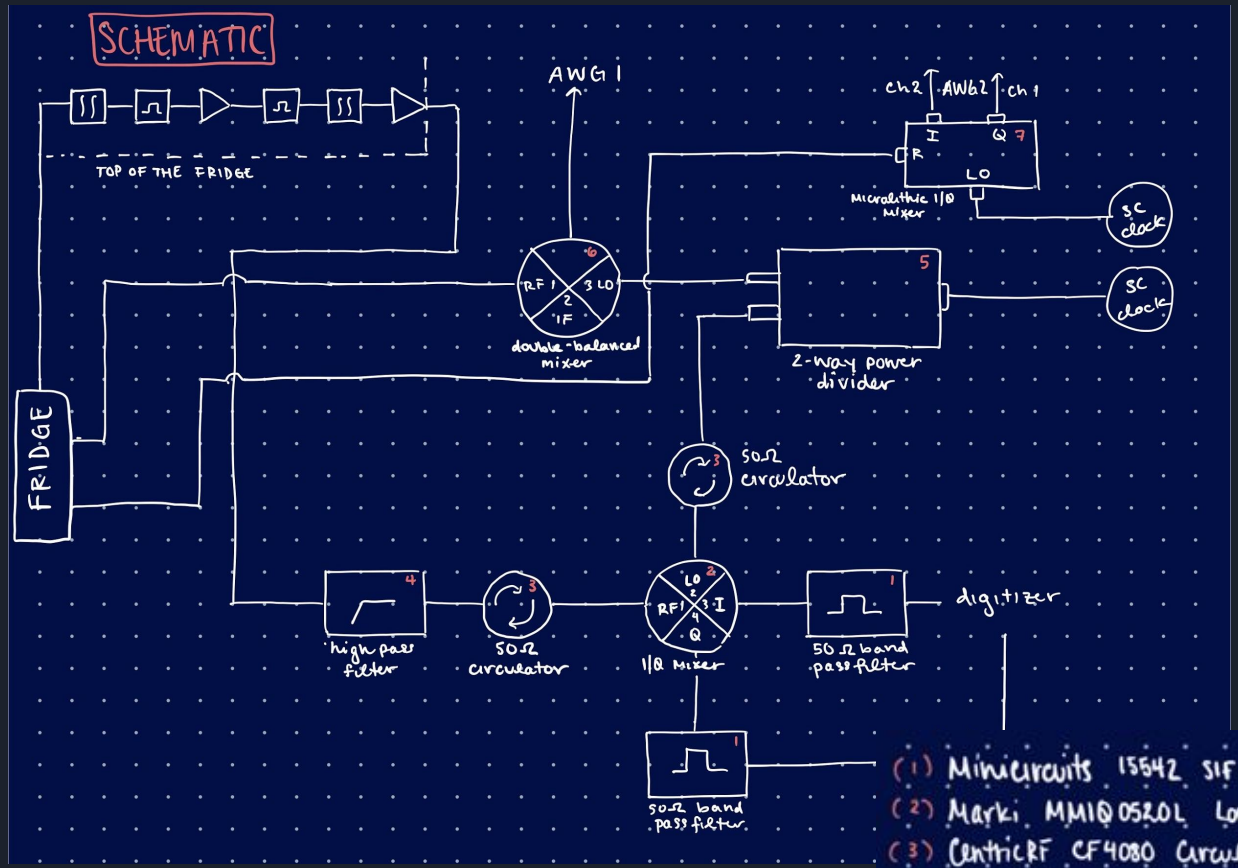
While I did get the correct answer the majority of the time, this noise model only had systematic noise with low probabilities. With higher circuit depths and stochastic noise, there is a higher probability of error.

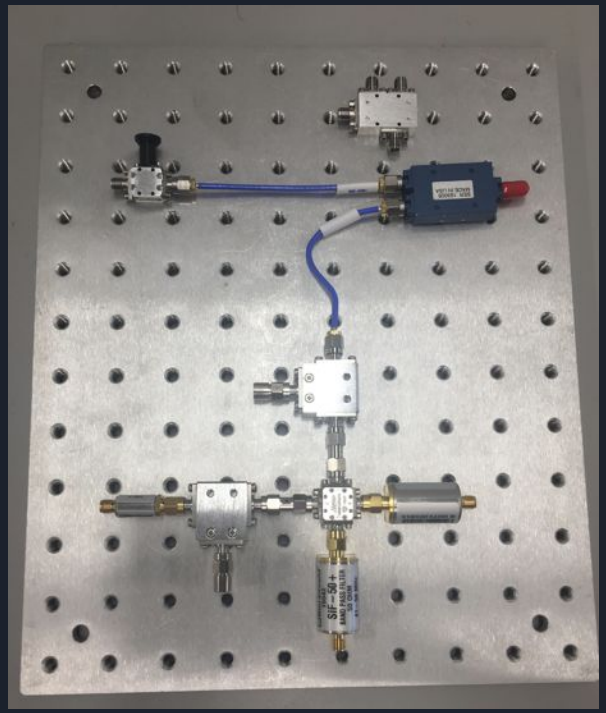
Quantum Error Detection

Thankfully, there are ways of detecting errors in quantum circuits, such as this circuit



MICROWAVE TABLE





(1) Minicircuits 15542 SIF-50+ Band Pass Filter 50ohm 41-58MHz
(2) Marki MMIQ0520L Low LO Drive Passive GaAs MMIC I/Q Mixer
(3) CentricRF CF4080 Circulator 4-8GHz VSWR 1.35 S Steel SMA 50Watts
(4) Minicircuits 15542 VHF-3500+ High Pass Filter 3400-9800 MHz
(5) Krytar Model 6020080 2-way Power Divider 2-8GHz
(6) Marki MM1D320L GaAs Double-Balanced Mixer
(7) Marki MLIQD218L Microlithic Double-Balanced I/Q Mixer

Planned Layout

BAND PASS FILTER (1)

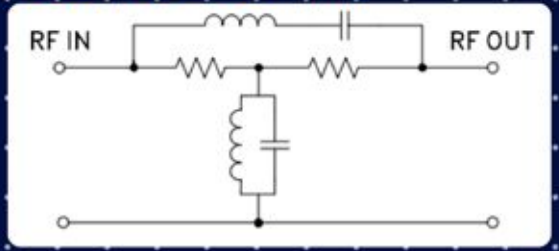
Maximum Ratings

- Operating & Storage Temperature: -55°C to 100°C
- RF power input: 0.5W

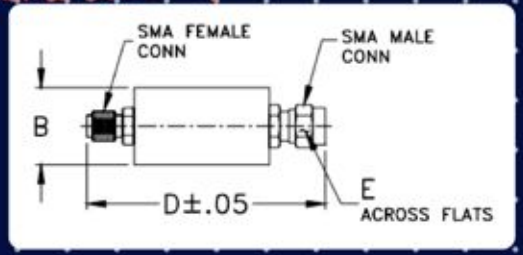
Electrical Specifications:

- Center Frequency: 50MHZ
- Passband (loss < 1dB): 41-58 MHZ
- Stopbands (loss > 10dB) at MHz: 11.5 & 200
  (loss > 20dB) at MHz: 3.1 & 350
- VSWR, 1.3:1 Typ. Total Band: DC-440 MHZ

electrical schematic



outline drawing



About:

- Used to isolate or filter out certain frequencies within a particular band (range).
- Passes signals within a certain band without distorting the input signal or introducing noise
  - Known as bandwidth
- amplitude of output signal always less than input signal.

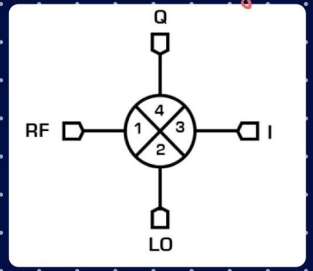
I/Q MIXER (2)

LO: local oscillator
RF: radio frequency

Maximum Ratings

- Operating Temperature: -55° to 100°C
- Storage Temperature: -65° to 125°C
- Power Handling: +23 dBm
- Port 3 DC current: 50mA
- Port 4 DC current: 50mA

Functional Block Diagram



Electrical Specifications:

- RF frequency range: 5-20 GHz
- LO frequency range: 5-20 GHz
- I frequency range: 0-6 GHz
- Q frequency range: 0-6 GHz
- Amplitude balance: 0.4dB
- Phase balance: 2°

About:

- Creates new frequencies from two frequencies applied to it
- Two mixers - RF/LO connected to in-phase power divider, opposite connected w/ quadrature hybrid.
- I (in-phase) and Q (90° out-of-phase) components for user
- fundamental principle is signal cancellation through phase manipulation
- Create two in-phase copies of desired signal & two out-of-phase copies of undesired signal.

CIRCULATOR (3)

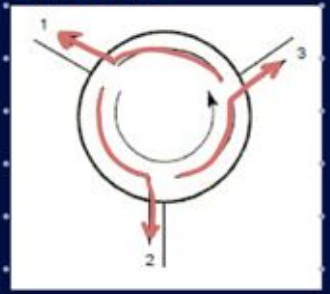
Electrical Specifications:

- Frequency range: 4-8GHz

About:

- Passive RF device, made of magnets & ferrite material
- controls direction of signal flow
- Y-junction. works by cancelling waves propagating over two paths near magnetized material

Schematic

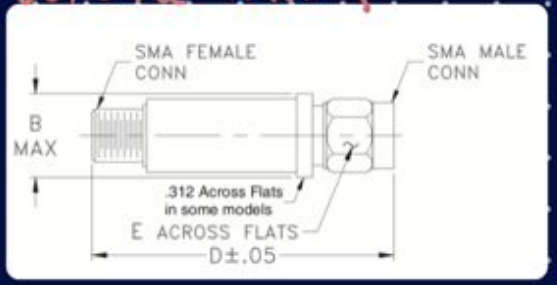


HIGH PASS FILTER (4)

Maximum Ratings:

- Operating & storage temperature: -55° to 100° C
- RF Power input: 7W @ 25°C (linearly derate to 3W @ 100°C)

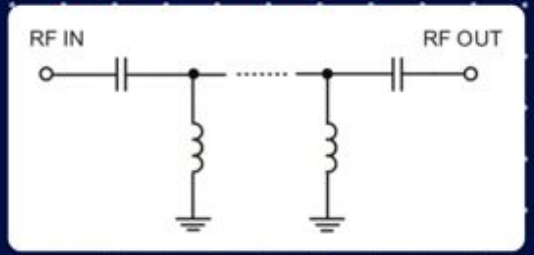
Outline drawing



Electrical Specifications:

- Stopband (loss > 20dB): DC - 2800 min.
- Passband (loss < 2dB): 3900 - 9800 Max
- VSWR: 3650 - 9500 MHz

Electrical Schematic



About:

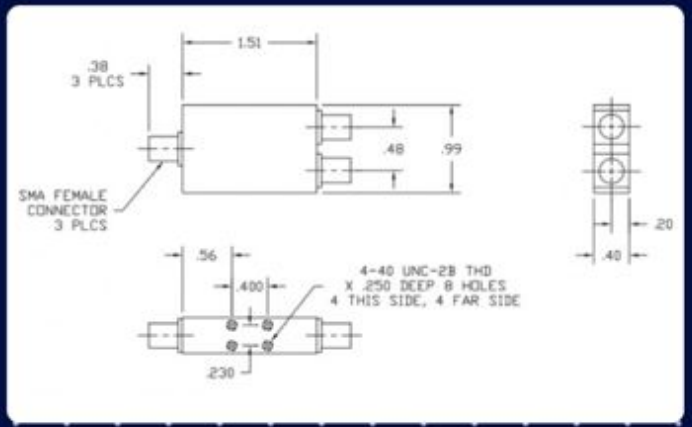
- Only allows frequencies above cutoff point.

POWER DIVIDER (5)

Features:

- Frequency Operation: 2-8 GHz
- Isolation: >19.5dB
- Insertion loss: <0.8dB
- Amplitude tracking: ±0.25dB
- Max VSWR: 1.45 GHz

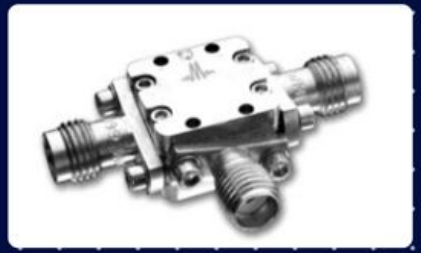
outline drawing



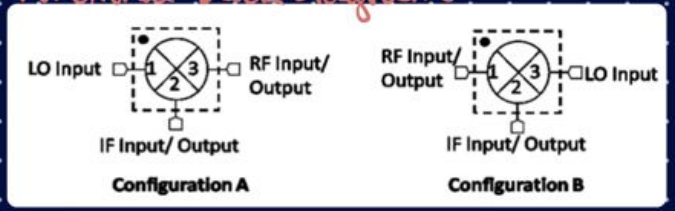
About:

- Splits input signal into two outputs equal in amplitude & phase.
- Wants to maximize port-to-port isolation, minimize insertion loss & VSWR

DOUBLE-BALANCED MIXER (6)



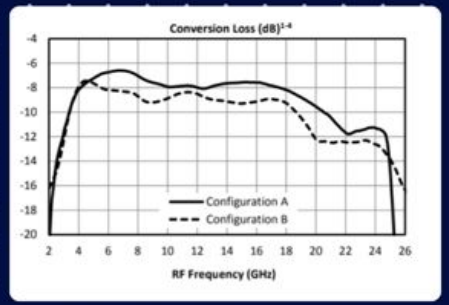
Functional Block Diagram

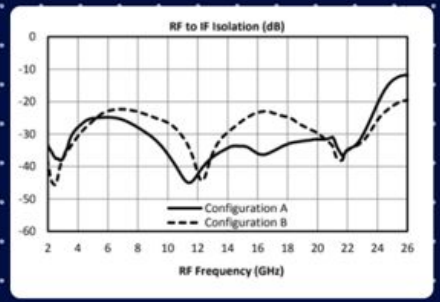


Absolute Maximum Ratings:

- Port 1 DC current: 15mA
- Port 2 DC current: 30 mA
- Port 3 DC current: 15 mA
- Operating temperature: -55° to 100°
- Storage temperature: -66° to 125°
- RF power handling: +25dBm at 25° C, derated linearly to 21 dBm at 100°

Electrical Specifications





MICROLITHIC DOUBLE-BALANCED I/Q MIXER

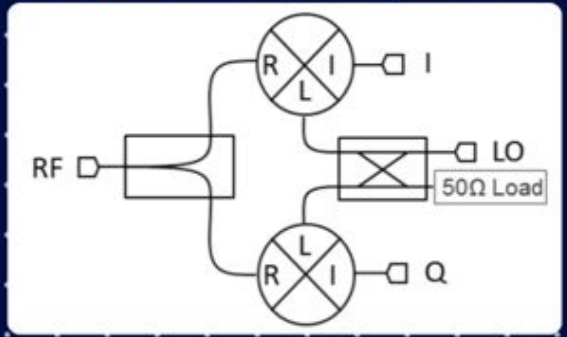
Electrical Specifications:

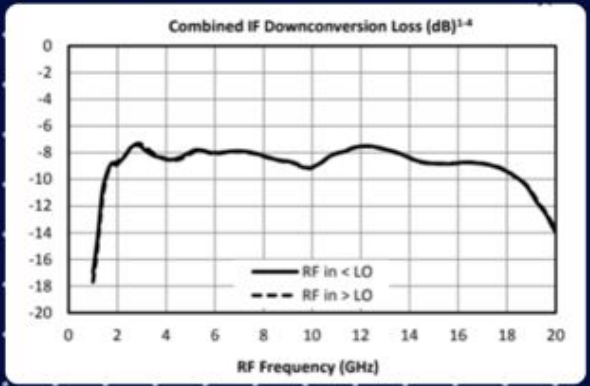
- Frequency range: 2-18 GHz

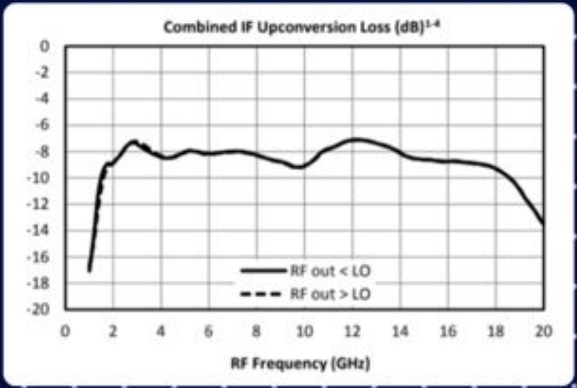
About:

- Matched double-balanced mixers connected w/ integrated LO hybrid & RF power divider
- Used for upconversion or downconversion

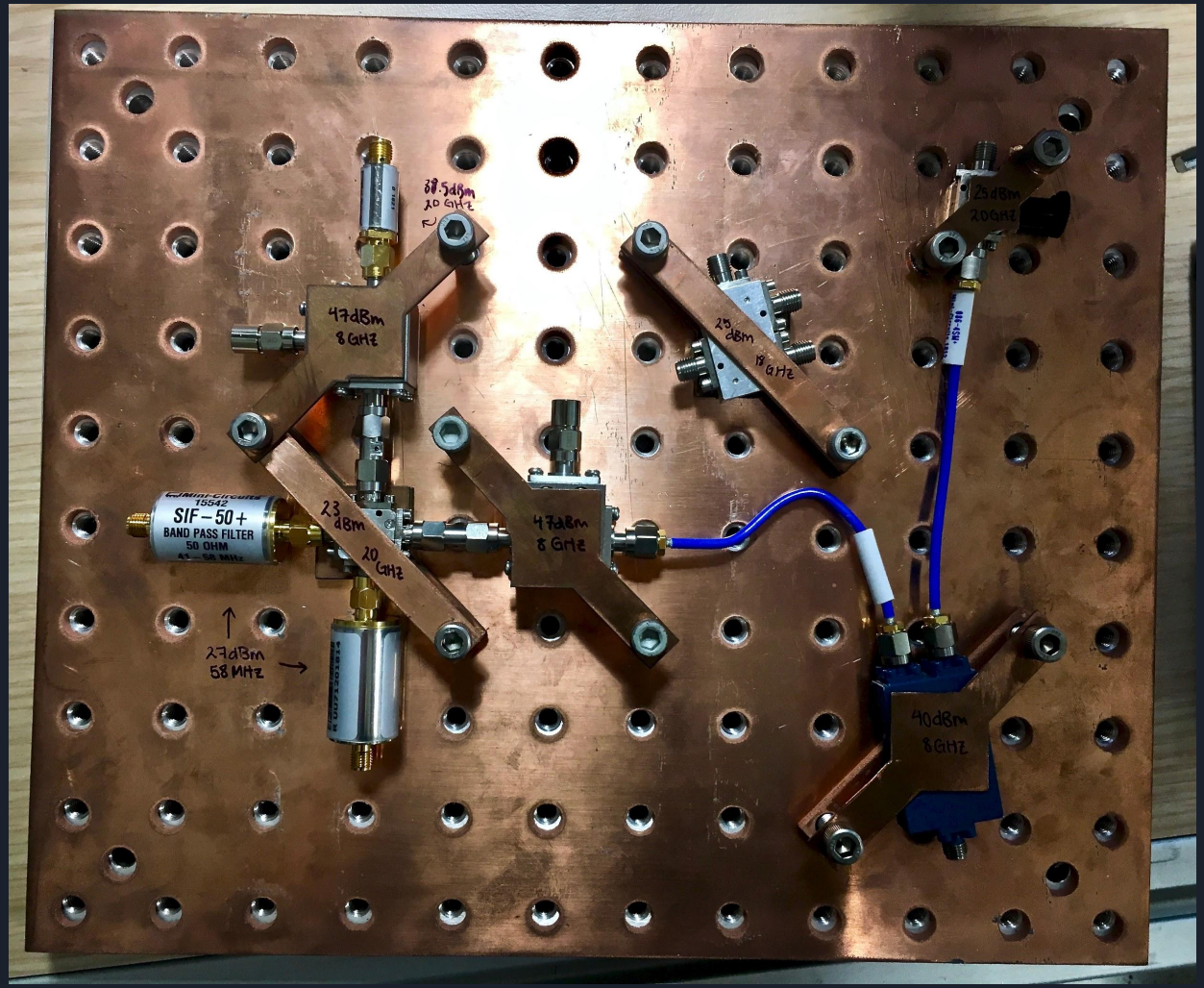
Functional Block Diagram







Element	Max Power	Max Frequency
Band Pass Filter (1)	27 dBm	58 MHz
I/Q Mixer (2)	23 dBm	20 GHz
Circulator (3)	47 dBm	8 GHz
High Pass Filter (4)	38.5 dBm	20 GHz
Power Divider (5)	40 dBm	8 GHz
Double-Balanced Mixes (6)	25 dBm	20 GHz
Microlithic DB Mixes (7)	25 dBm	18 GHz


38.5dBm
20 GHz
47dBm
8 GHz
25dBm
20GHz
23 dBm
8 GHz
Mini-Circuits
15542
SIF-50+
BAND PASS FILTER
50 OHM
23
dBm
20
GHz
47dBm
8 GHz
27dBm
58 MHz
40dBm
8 GHz

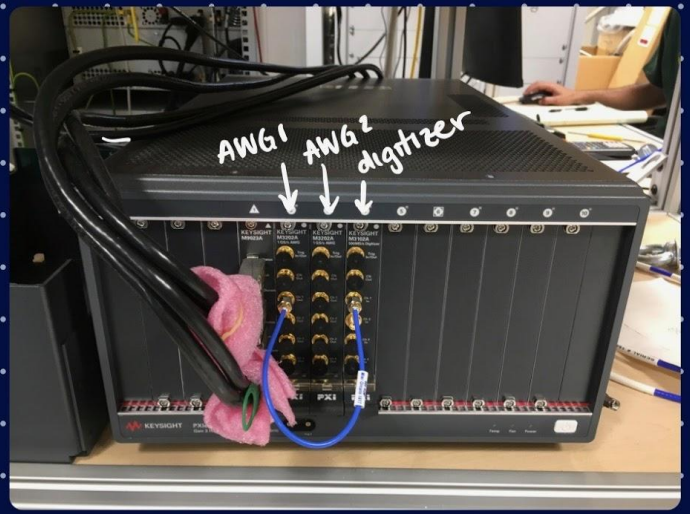
Final Setup

SIGNAL SOURCE

Signal Core SC5511A

- synthesized signal generator with output frequency range 100MHz to 20GHz

control screen



AWG & DIGITIZER

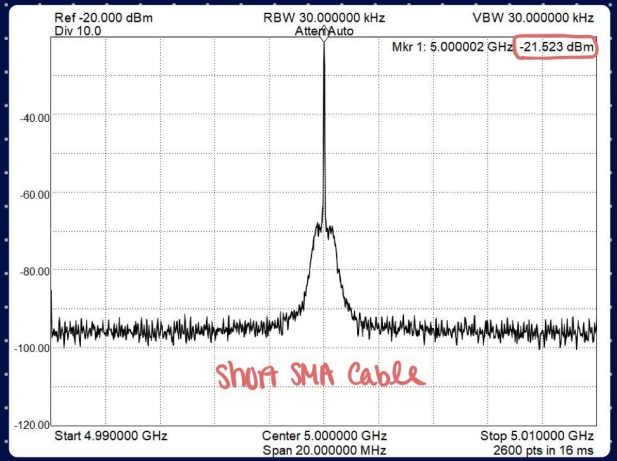
Keysight M3202A PXIe Arbitrary Waveform Generator
- 1 GSa/s (GSa/s = Giga Samples/second, number of times per second the waveform is measured
- 14bit
- 400 MHz bandwidth

Keysight M3102A PXIe Digitizer
- 500 MSa/s
- 14 bit
- 200 MHz bandwidth

SPECTRUM ANALYZER Signal Hound USB-SA124B

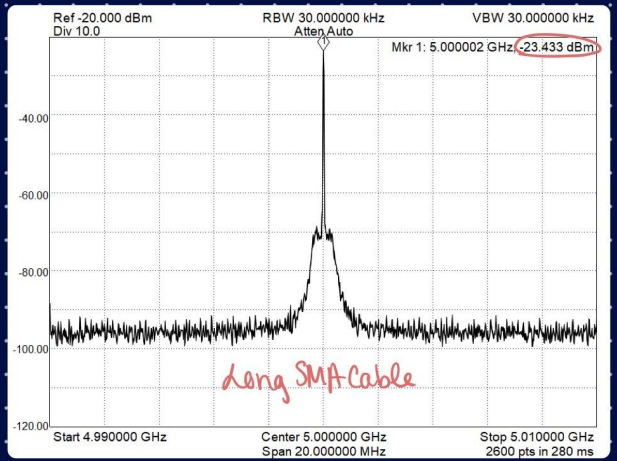
- 12.4 GHz Software Defined Radio
- -151 dBm to +10 dBm

FIRST TESTS : attenuation

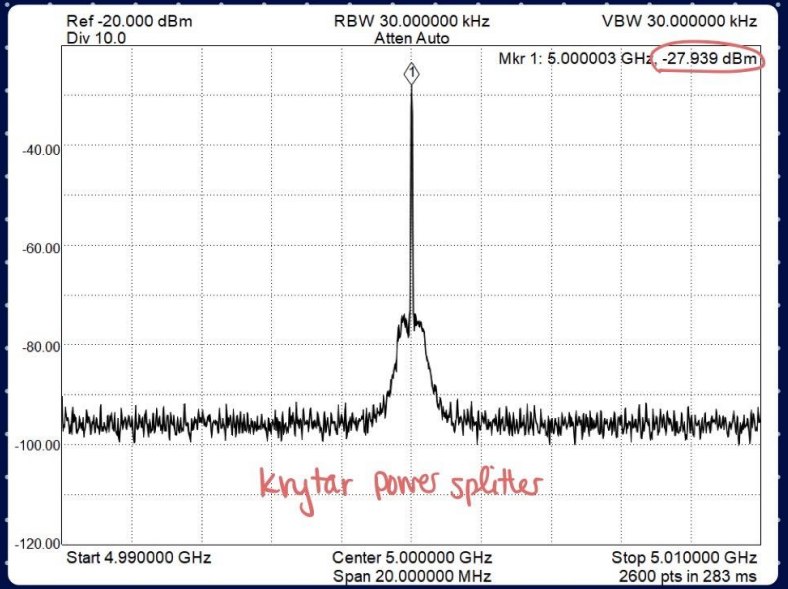


-20dBm attenuator

-21.523
+20.000
-1.523 attenuation from short SMA



-23.433
+20.000
-3.433 attenuation from long SMA

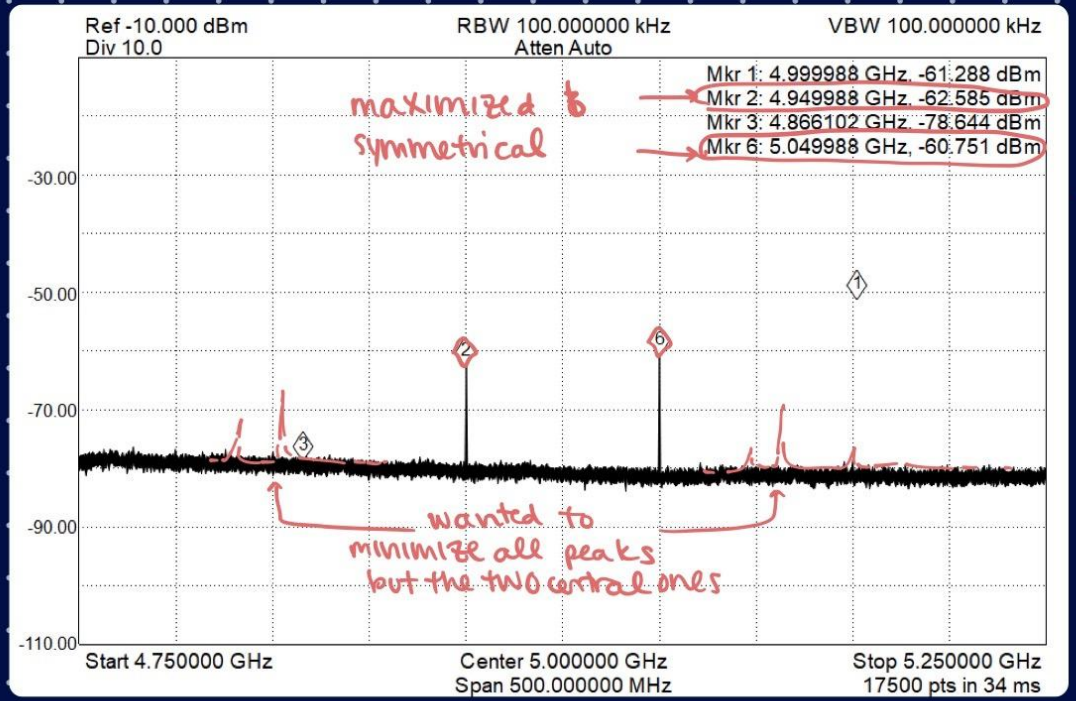


- 27.939 (avg. channel 1 & channel 2)
+ 20.000 (attenuator)
+ 3.433 (long cable)

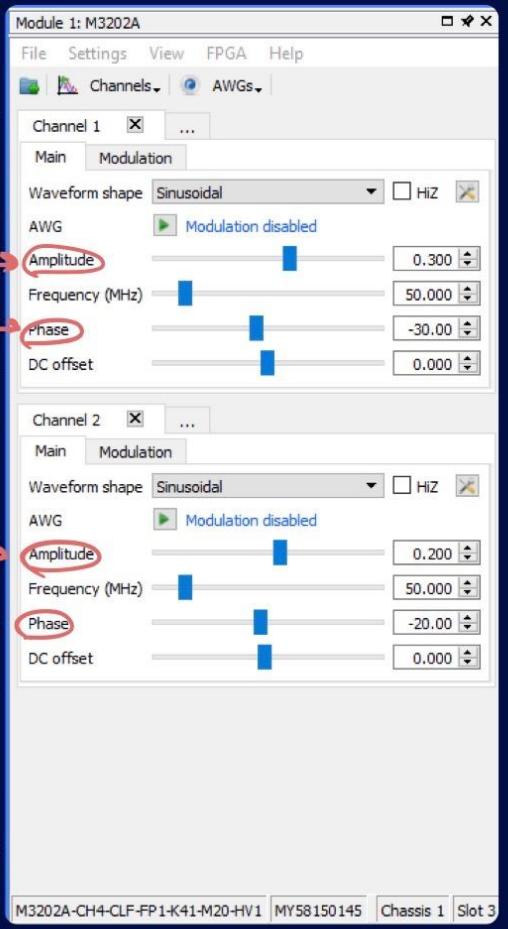
4.506

↑ attenuation from power splitter

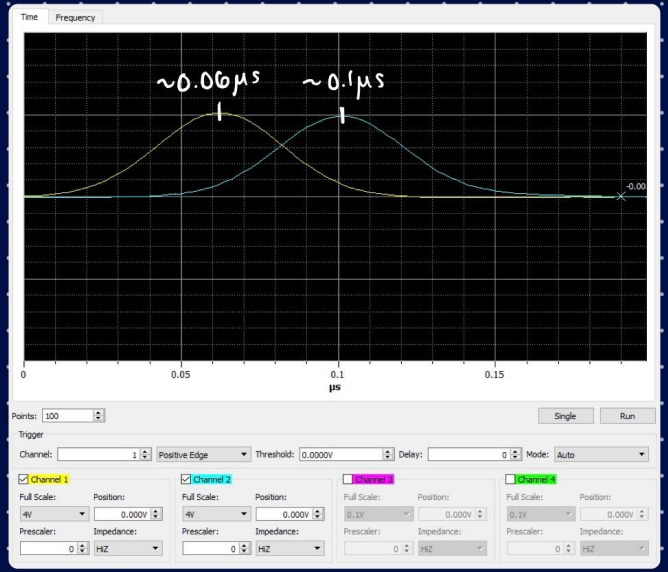
SECOND TEST - Microlithic I/Q Mixes



tuned these variables to accomplish this (trial & error)



FINAL TEST - Microlithic I/Q Mixer



First, calculated amplitude:

max power = 25 dBm, chose 15dBm

15dBm = 0.032 Watts

$0.032W = \frac{V^2}{50\Omega}$

$1.6 = V^2$

$\sqrt{1.6} = V$

$1.27 = V$ (amplitude)

seek velocity of signal: $0.1\mu s - 0.06\mu s = 40ns$

$\ell_{long\ cable} = 20ft$

$\ell_{short\ cable} = 8in$

$\rightarrow 240in - 8in = 232in$

$40ns = 4.0 \cdot 10^{-9}$ seconds

$232in = 5.89$ meters

$v = \frac{\ell}{t} = \frac{5.89}{4.0 \cdot 10^{-9}} = 1.47 \cdot 10^9\ m/s$