NYU

Summer Q Camp 2020: Final Presentation

Shabani Lab

Bloomberg

wp.nyu.edu/shabanilab

QCamp Structure:

- **Quantum Programming (Qiskit)**
  - **Quantum Masters: Algorithms**
  - **Quantum Troopers: Wavefunction Generation as Noise Evaluation**
  - **Quantum Wizards: Financial Modeling - Option Pricing**
- **Quantum Information Science**
  - **Quantum Masters**
- **Experimental Condensed Matter**
  - **Quantum Guardians: Automated Graphene Exfoliation**
- **Advanced Topics**
  - **Quantum-Plators: TQC + Circuit QED Lattices**
  - **Quantum Wizards: Quantum Hardware**

Modular Exponentiation and the Hard Problems

Eva Gurra, Lucy Jiao

Motivation - AKA the Hard Problem

- Public Key Cryptography: talking over public networks
- Cryptography relies on certain problems being computationally hard
  - Exponential scaling in time
- Shor's algorithm threatens some of the most popular schemes (RSA and elliptic curve cryptography)
  - Solves the previously hard problems of prime factorization or discrete logarithms in polynomial time
  - Key step : Controlled Modular Exponentiation

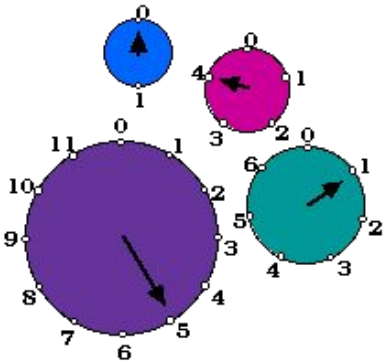
Modular Arithmetic

The modulo operation finds the remainder after division of some number by the modulus:

$$5 \; mod \; 2 = 1 \rightarrow (2 \times 2) + 1$$

Modulus

Remainder



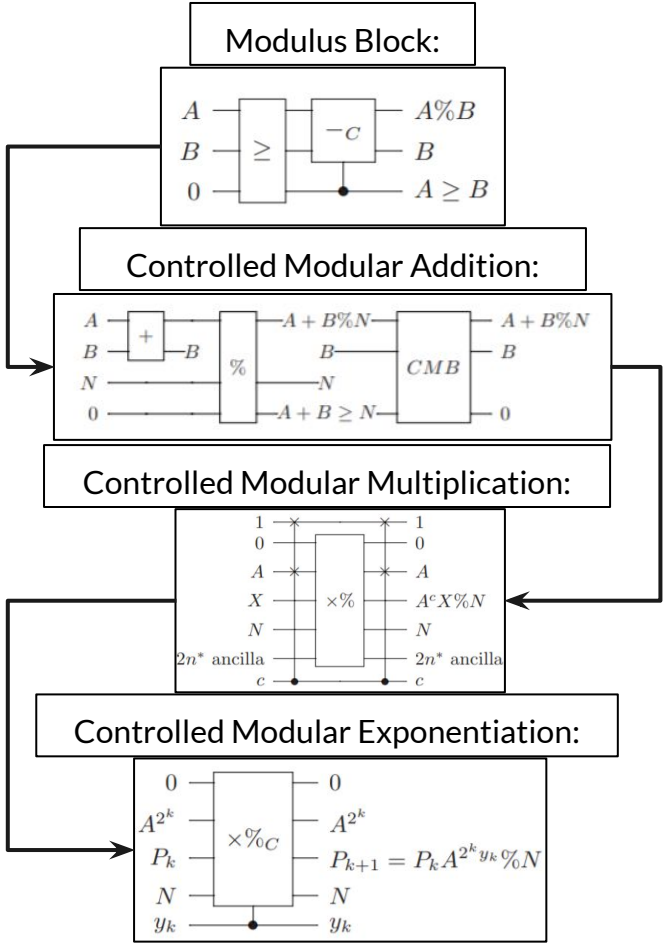
Congruence: For a positive integer *n*, integers *a* and *b* are **congruent** mod *n*, if their remainders are the same when divided by *n*.

$$9 \equiv 4 \; (mod \; 5)$$

Modular Exponentiation - General

$$f(x) = a^x \ mod \ N$$

- **Modular Addition**
  - add two numbers and take the modulus
- **Modular Multiplication**
  - Repeated use of modular addition and modular doubling
- **Modular Exponentiation**
  - Repeated use of Modular Multiplication



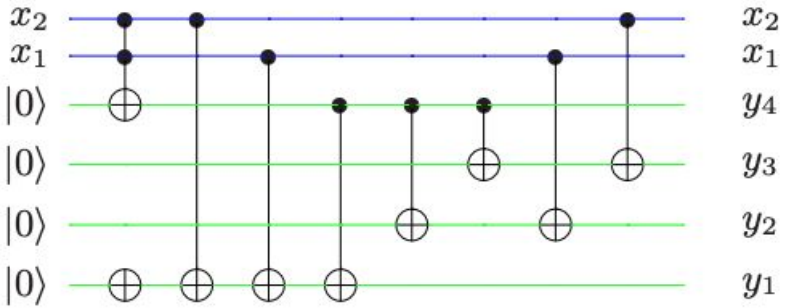
Modular Exponentiation - Specific N

1. Create a truth table for the binary representation of a^x mod N
2. Use CNOT gates to manipulate the state to match the truth table
   a. i.e. 00 -> 01, 01 -> 10
   b. This becomes more complex as we deal with larger numbers

a x N

```
binaryTruthTable(2, 4, 15)
```

x_2	x_1	y_4	y_3	y_2	y_1
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Gamel and James (2013)

Shor's Algorithm

- Factoring algorithm for N=pq where p,q are prime

1. Pick an { *a* } such that *a* is coprime with N , i.e. gcd(*a*,N) =1
2. Find the smallest { r } such that $a^r = 1 \bmod N$
    a. This is the "period" of the function
3. If r is even
    a. Let $x = a^{r/2} \bmod N$, check $x+1 \neq 0 \bmod N$
    b. Then p,q = gcd(x+1, N), gcd(x-1, N)
    c. Then we are done
4. If r is odd: pick a different *a* and do it again

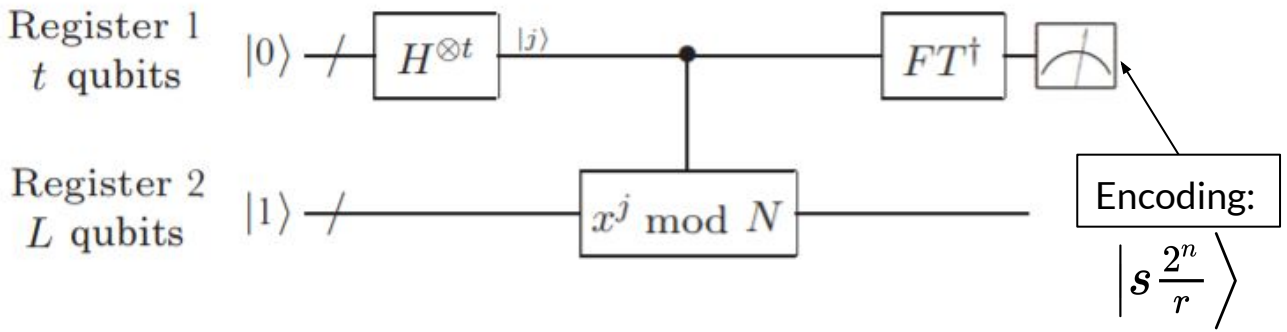
This is the order-finding step allowing for a "quantum speed-up"

Quantum Order Finding

Given: $f(x) = a^x \ mod \ N$

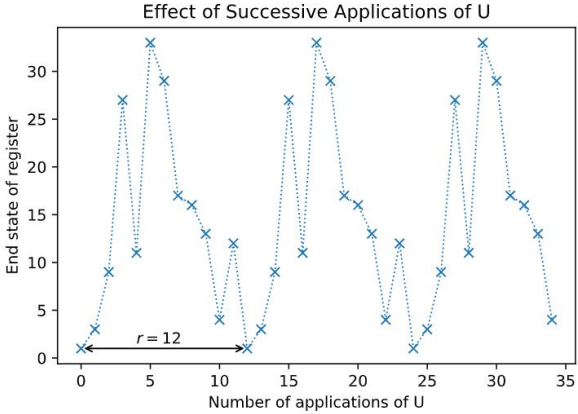
Problem: Determine the smallest non-zero integer, *r*, such that $a^r \ mod \ N = 1$

Solution: Build a Unitary operator to represent f(x) and apply Quantum Phase Estimation.



Example: N = 35, a = 3

$$f(x) = 3^x \ mod \ 35$$



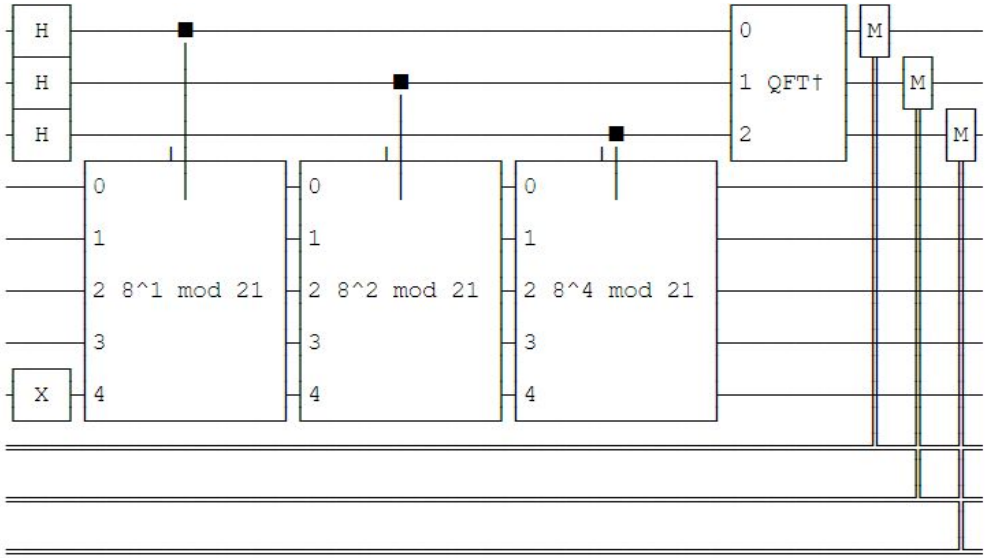
$$U|1\rangle = |3\rangle$$
$$U^2|1\rangle = |9\rangle$$
$$U^3|1\rangle = |27\rangle$$
$$\vdots$$
$$U^{(r-1)}|1\rangle = |12\rangle$$
$$U^r|1\rangle = |1\rangle$$

Factoring 21

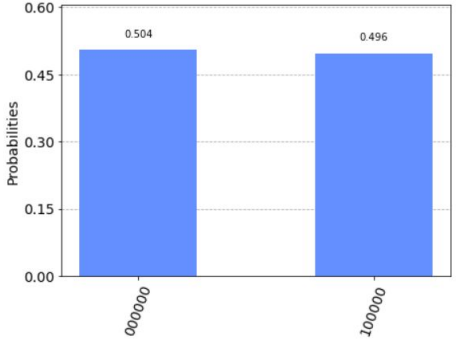
$$f(x) = 8^x \ mod \ 21$$

```
binaryTruthTable(8, 8, 21)
```

x_3	x_2	x_1	y_5	y_4	y_3	y_2	y_1
0	0	0	0	0	0	0	1
0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	1
0	1	1	0	1	0	0	0
1	0	0	0	0	0	0	1
1	0	1	0	1	0	0	0
1	1	0	0	0	0	0	1
1	1	1	0	1	0	0	0




```
Register Reading: 100
Corresponding Phase: 0.500000
Result: r = 2
Guessed Factors: 7 and 3
*** Non-trivial factor found: 7 ***
*** Non-trivial factor found: 3 ***
```



Factoring 51

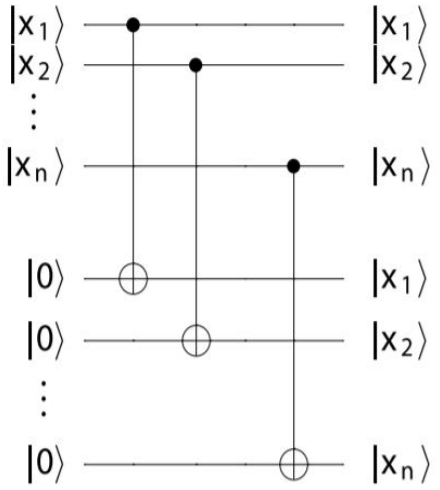
- Fermat Primes are special primes that have the form:

$$F_n = 2^{2^n} + 1$$

- Sequence: 3, 5, 17, 257…
- 51 = 3 x 17, a product of Fermat primes
- This circuit can detect periods that are powers of 2.

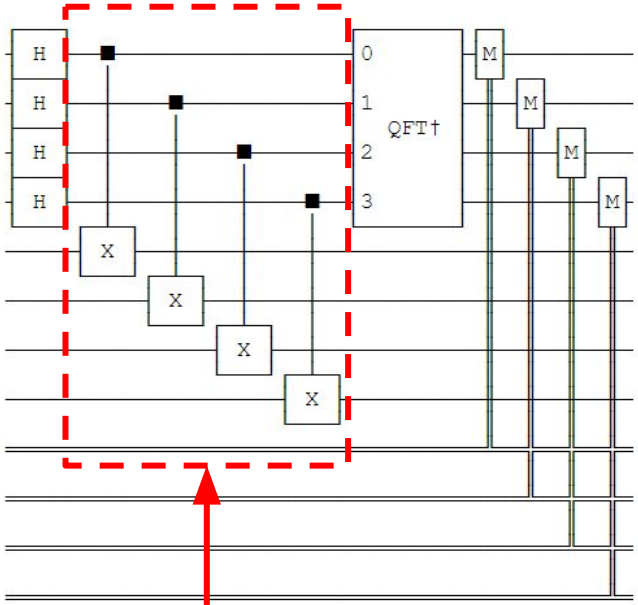
Compressed Modular Exponentiation:

$$a^x \bmod N \rightarrow x \bmod r(a)$$

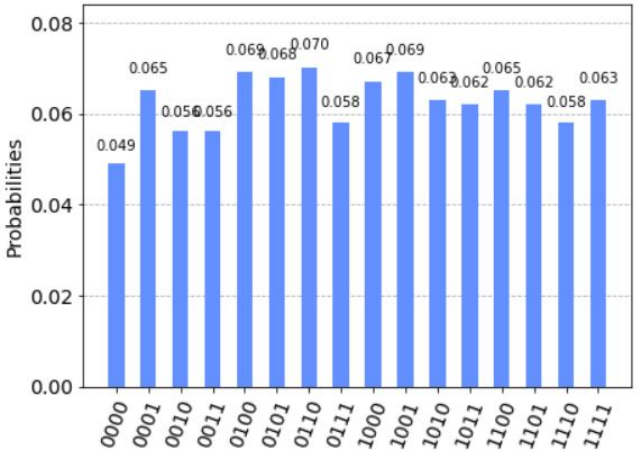


$$|x\rangle \otimes |0 \cdots 0\rangle \rightarrow |x\rangle \otimes |x \bmod \mathrm{r}\rangle$$

Factoring 51



Compressed Modular Exponentiation

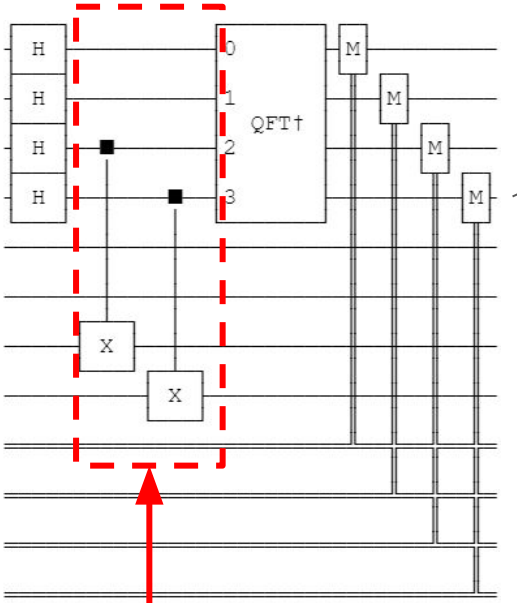


Post-Processing:

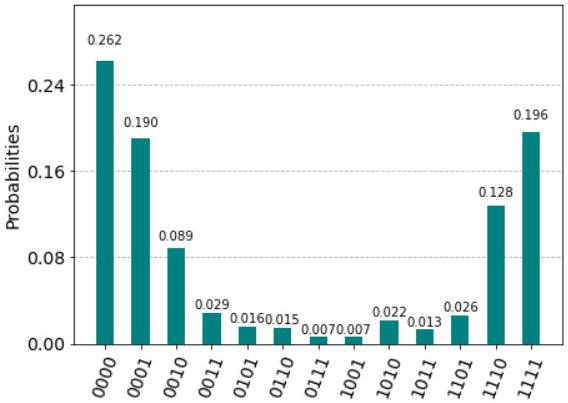
```
[[0.6875, '11/16', 16],
 [0.1875, '3/16', 16],
 [0.8125, '13/16', 16],
 [0.125, '1/8', 8],
 [0.0, '0/1', 1],
 [0.625, '5/8', 8],
 [0.4375, '7/16', 16],
 [0.375, '3/8', 8],
 [0.5625, '9/16', 16],
 [0.3125, '5/16', 16],
 [0.0625, '1/16', 16],
 [0.75, '3/4', 4],
 [0.5, '1/2', 2],
 [0.9375, '15/16', 16],
 [0.25, '1/4', 4],
 [0.875, '7/8', 8]]
```

```
Register Reading: 1001
Corresponding Phase: 0.562500
Result: r = 16
Guessed Factors: 3 and 17
*** Non-trivial factor found: 3 ***
*** Non-trivial factor found: 17 ***
```

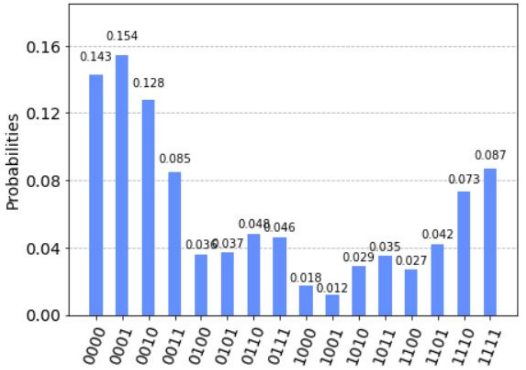
Factoring 51



Compressed Modular Exponentiation



Bit Flip Error:




```
Register Reading: 1100
Corresponding Phase: 0.750000
Result: r = 4
Guessed Factors: 3 and 17
*** Non-trivial factor found: 3 ***
*** Non-trivial factor found: 17 ***
```

Conclusions and Future Research

- Current implementations of Shor's algorithm require knowledge of the period in order to build a modular exponentiation circuit and this method is NOT scalable.
- Finding an optimized solution for the general modular exponentiation gate
  - Provides practical realization of Shor's algorithm
  - Add general functionality to Qiskit
    - Needs modular addition and multiplication
- Implement the algorithm for discrete log implementations and Elliptic curves

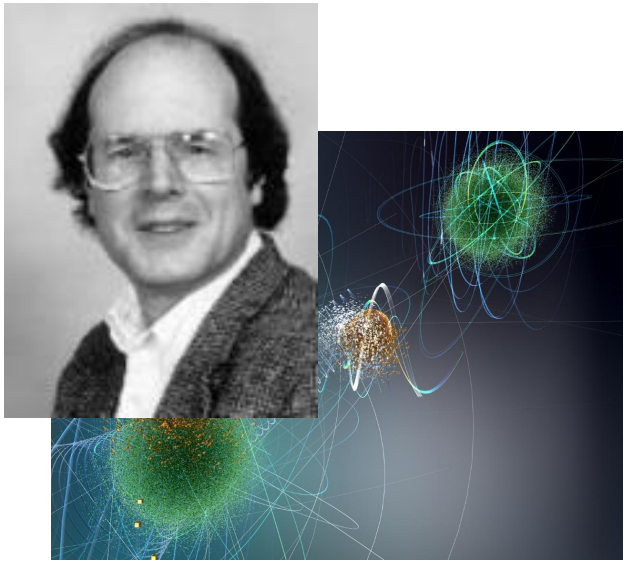
References

1. Gamel, Omar, and Daniel FV James. "Simplified Factoring Algorithms for Validating Small-Scale Quantum Information Processing Technologies." *arXiv preprint arXiv:1310.6446* (2013).
2. Nielsen, M. A., & Chuang, I. L. (2010). Chapter 5: Quantum Fourier Transform and its Applications. In *Quantum Computation and Quantum Information* (pp. 216-240). Cambridge: Cambridge University Press.
3. Geller, M., Zhou, Z. Factoring 51 and 85 with 8 qubits. *Sci Rep* 3, 3023 (2013). https://doi.org/10.1038/srep03023
4. Gamel, Omar, and Daniel FV James. "Synthesizing quantum circuits for simple periodic functions for quantum information and computation." *Quantum Information & Computation* 14.9&10 (2014): 763-776.
5. Markov, Igor L., and Mehdi Saeedi. "Constant-optimized quantum circuits for modular multiplication and exponentiation." *arXiv preprint arXiv:1202.6614* (2012).
6. Qiskit Textbook (https://qiskit.org/textbook/preface.html)

Quantum Teleportation

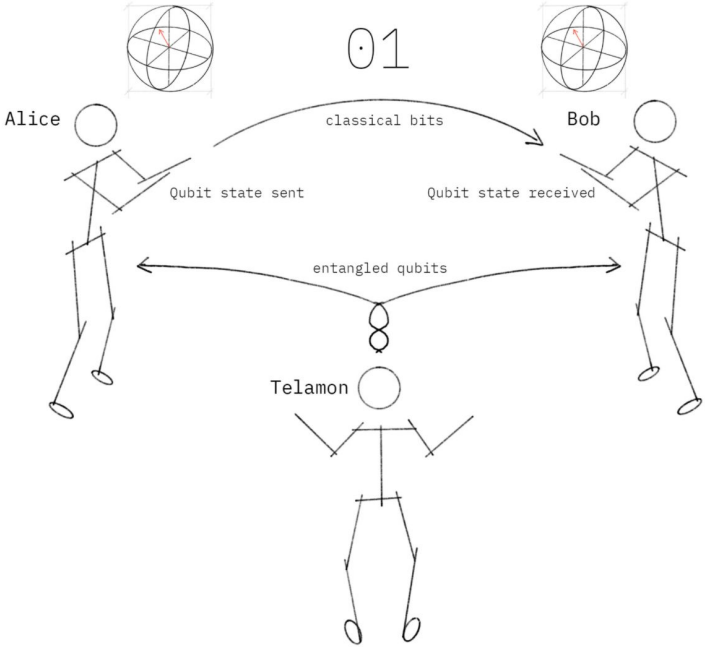
Esther Bistricer and Elena Lukac

Background



- Proposed theoretically in 1993
- Charles Bennett named this “quantum teleportation” instead of “telepheresis”, its former name
- Transfer of a particle’s quantum state onto another particle, while erasing the state in the original particle
- Allows for secure messaging between two parties

Algorithm



We can think of this algorithm as a **transfer of information** between two people, Alice and Bob.

Example: $|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$

Qiskit Code Implementation: Set Up

- Import all packages
- Create quantum circuit

```
# Do the necessary imports
import numpy as np
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, execute, BasicAer, IBMQ
from qiskit.visualization import plot_histogram, plot_bloch_multivector
from qiskit.extensions import Initialize
from qiskit_textbook.tools import random_state, array_to_latex
```

```
qr = QuantumRegister(3)     # Protocol uses 3 qubits
crz = ClassicalRegister(1) # and 2 classical bits
crx = ClassicalRegister(1) # in 2 different registers
teleportation_circuit = QuantumCircuit(qr, crz, crx)
```

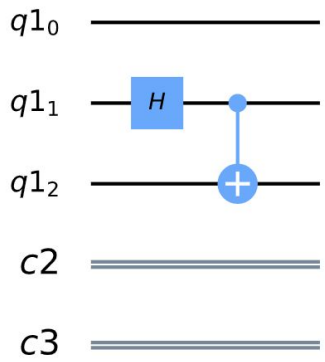
Qiskit Code Implementation: Step 1

- Create entangled pair of qubits (Bell pair)
  - Apply Hadamard gate to one qubit, then apply CNOT gate to other qubit controlled by first qubit

```
def create_bell_pair(qc, a, b):
    """Creates a bell pair in qc using qubits a & b"""
    qc.h(a) # Put qubit a into state |+>
    qc.cx(a,b) # CNOT with a as control and b as target
```

```
## SETUP
# Protocol uses 3 qubits and 2 classical bits in 2 different registers
qr = QuantumRegister(3)
crz, crx = ClassicalRegister(1), ClassicalRegister(1)
teleportation_circuit = QuantumCircuit(qr, crz, crx)

## STEP 1
# In our case, Telamon entangles qubits q1 and q2
# Let's apply this to our circuit:
create_bell_pair(teleportation_circuit, 1, 2)
# And view the circuit so far:
teleportation_circuit.draw()
```

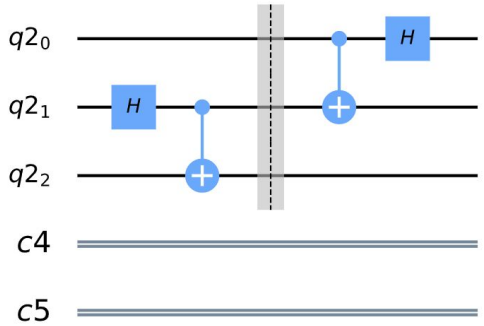


Qiskit Code Implementation: Step 2

- Apply CNOT gate to q_1 controlled by $|\psi\rangle$
- Apply Hadamard gate to $|\psi\rangle$

```
def alice_gates(qc, psi, a):
    qc.cx(psi, a)
    qc.h(psi)
```

```
## STEP 2
teleportation_circuit.barrier() # Use barrier to separate steps
alice_gates(teleportation_circuit, 0, 1)
teleportation_circuit.draw()
```

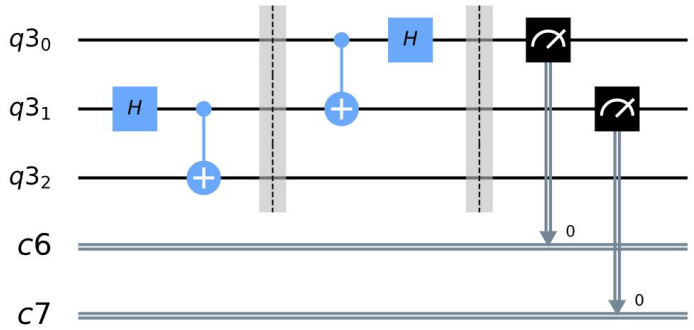


Qiskit Code Implementation: Step 3

```
def measure_and_send(qc, a, b):
    """Measures qubits a & b and 'sends' the results to Bob"""
    qc.barrier()
    qc.measure(a,0)
    qc.measure(b,1)
```

```
## STEP 3
measure_and_send(teleportation_circuit, 0 ,1)
teleportation_circuit.draw()
```

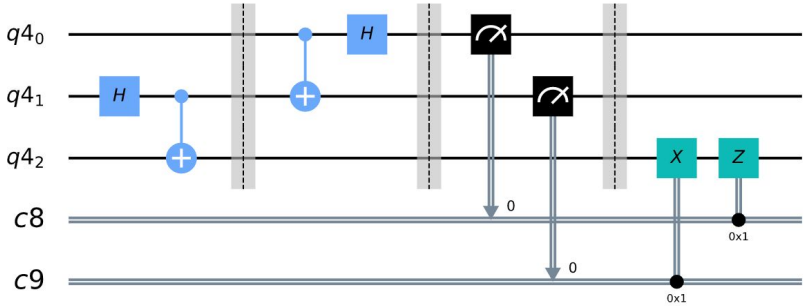
- Apply measurement to $q1$ and $|\psi\rangle$ and store in two classical bits
- "Send" bits



Qiskit Code Implementation: Step 4

```
# This function takes a QuantumCircuit (qc), integer (qubit)
# and ClassicalRegisters (crz & crx) to decide which gates to apply
def bob_gates(qc, qubit, crz, crx):
    # Here we use c_if to control our gates with a classical
    # bit instead of a qubit
    qc.x(qubit).c_if(crx, 1) # Apply gates if the registers
    qc.z(qubit).c_if(crz, 1) # are in the state '1'
```

```
## STEP 4
teleportation_circuit.barrier() # Use barrier to separate steps
bob_gates(teleportation_circuit, 2, crz, crx)
teleportation_circuit.draw()
```



- Depending on classical bits' state, apply following gates on q_2:

00	Do Nothing
01	Apply X gate
10	Apply Z gate
11	Apply ZX gate

- Teleportation complete

Our Summer Experience

This summer we learned about a variety of different topics including but not limited to:

- Introduction to linear algebra
- Quantum information
- Quantum teleportation
- Python
- What research in a lab can look like
- Different career paths that intersect with quantum
- How to independently study

Our Schedule

MONDAY: 8:30 am (Update Meeting)

TUESDAY: 1 pm (Update Meeting, Presentation)

WEDNESDAY: 8:30 am (Update Meeting), 9 am (Quantum Information Lesson)

THURSDAY: 10 am (Python/Qiskit Lesson), 2 pm (Whole Group Meeting)

FRIDAY: 8:30 am (Update Meeting), 9 am (Quantum Information Lesson)

Thank you...

Dr. Shabani, Billy Strickland, Bassel Heiba Elfeky, and everyone in the Quantum Masters group for making this summer full of learning and new experiences.

We would also like to acknowledge support from the Army Educational Outreach Program and our mentors from Bloomberg.

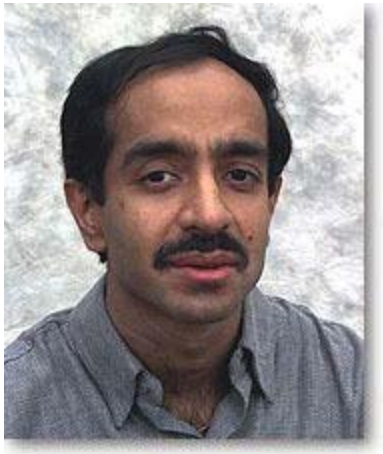
References

- https://qiskit.org/textbook/preface.html
- https://www.youtube.com/watch?v=fNk_zzaMoSs&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab
- https://www.geeksforgeeks.org/python-3-basics/?ref=lbp
- https://www.learnpython.org/
- https://www.youtube.com/watch?v=mMwovHK2NrE&index=5
- https://www.nature.com/news/quantum-teleportation-is-even-weirder-than-you-think-1.22321
- https://en.wikipedia.org/wiki/Quantum_teleportation#:~:text=Quantum%20teleportation%20is%20a%20process,the%20sending%20and%20receiving%20location.

Grover’s algorithm

Alex Harris - Ashkan Yazdi Zadeh

Lov Grover



Lov Kumar Grover is an American-Indian computer scientist, who’s originated the ‘Grover database search algorithm’ . After Shor’s algorithm, Grover's algorithm renown as the second major algorithm proposed for quantum computing.

Grover was ranked the 9th most prominent computer scientist from India.

Purpose

Grover's algorithm is a quantum algorithm used for searching an unsorted database with N entries. Imagine you have the list below, among all these properties there an item with a unique property where we wish to locate and here, we call that W.

Using classical computers we had to search N items (2^n). However using Grover's algorithm, we can find the marked items in $\sqrt{N}$ steps. Thus, this algorithm can be really time-saving and that's the reason of its significance.

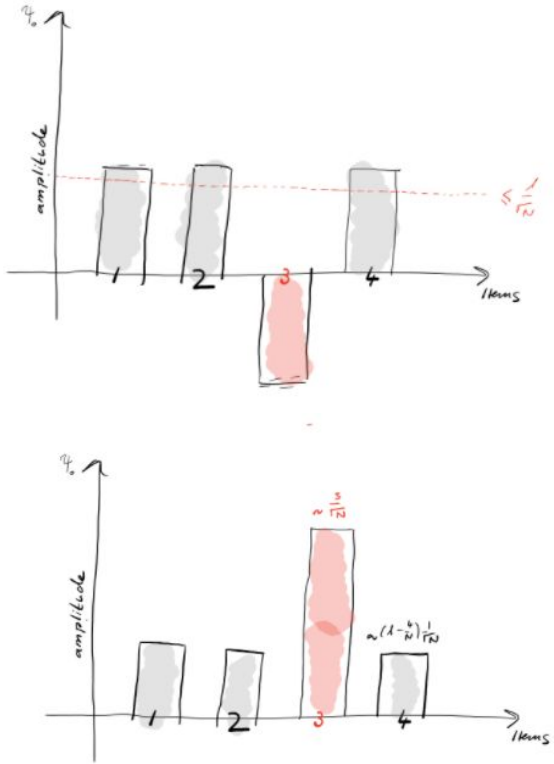
Purpose(continued)

Grover’s algorithm can also be used for other things such as:

- Estimating the mean and median of a set of numbers
- Solving the Collision problem
- Solving NP-problems by performing exhaustive searches over the set of possible solutions

Theory

- The oracle negates the amplitude of some basis state(s), and leaves the other amplitudes unchanged
- The diffuser then reflects all amplitudes about the average of the amplitudes
- After $\sqrt{N}$ operations, the qubit is measured, and the basis state(s) whose amplitude the oracle reflects are obtained with a high probability



Example—Qiskit Textbook

- Grover's Algorithm on 2 qubits to find $|00\rangle$
- Initialization
  - Hadamard on both qubits to create an equal superposition of all basis states

$$|\psi\rangle = \begin{bmatrix} 0.50000 \\ 0.50000 \\ 0.50000 \\ 0.50000 \end{bmatrix}$$

- Oracle
  - X gate on both qubits
  - Controlled Z gate
  - X gate on both qubits

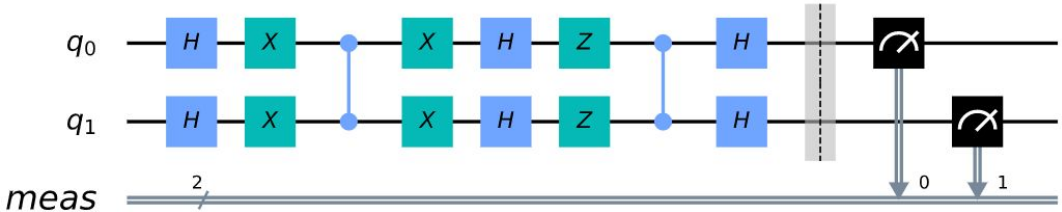
$$|\psi\rangle = \begin{bmatrix} -0.50000 \\ 0.50000 \\ 0.50000 \\ 0.50000 \end{bmatrix}$$

- Diffuser
  - Hadamard on both qubits
  - Z gate on both qubits
  - Controlled Z gate
  - Hadamard on both qubits

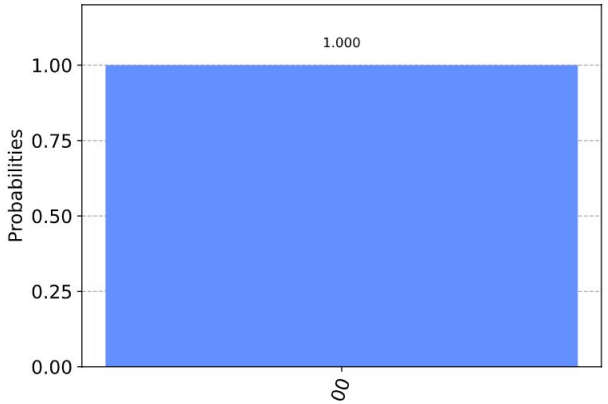
$$|\psi\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Example Continued

- Circuit Diagram



- Results

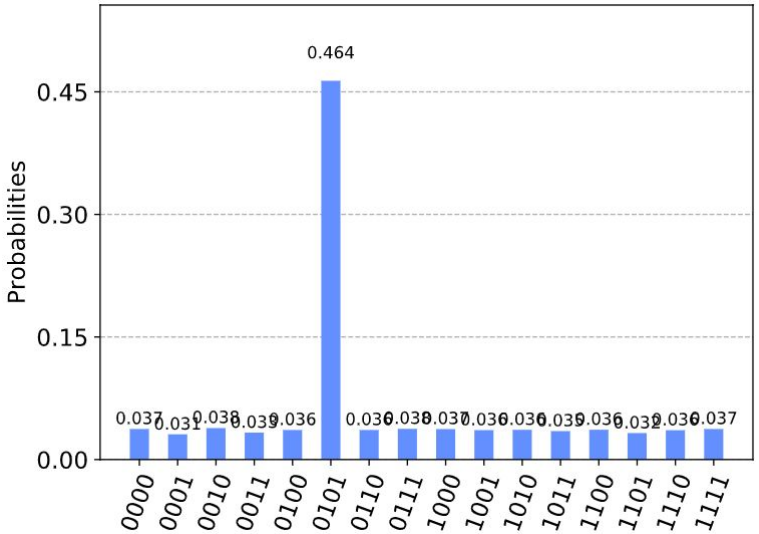


4-Qubit Implementation on Qiskit

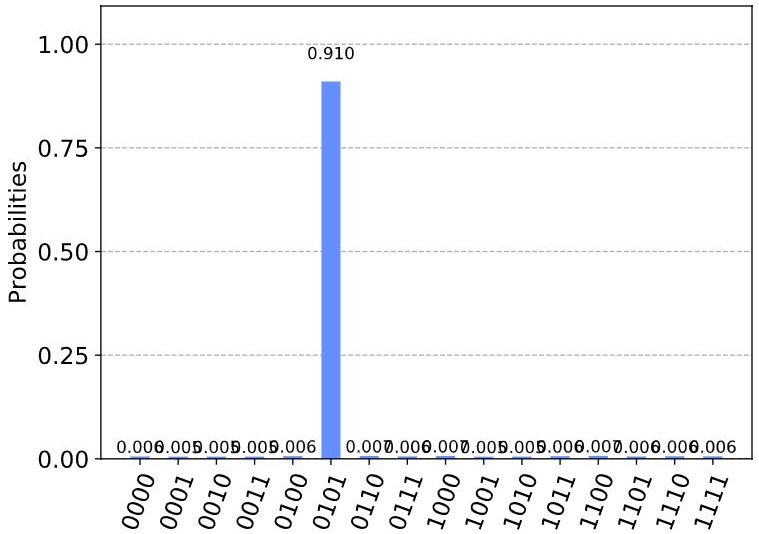
- Finding one marked 4-qubit state $|0101\rangle$
- Oracle
  - X gate on qubits 1 and 3
  - Triple-controlled Z gate
  - X gate on qubits 1 and 3
- Diffuser
  - Hadamard on all qubits
  - X gate on all qubits
  - Triple-controlled Z gate
  - X gate on all qubits
  - Hadamard on all qubits

Results

2 Operations

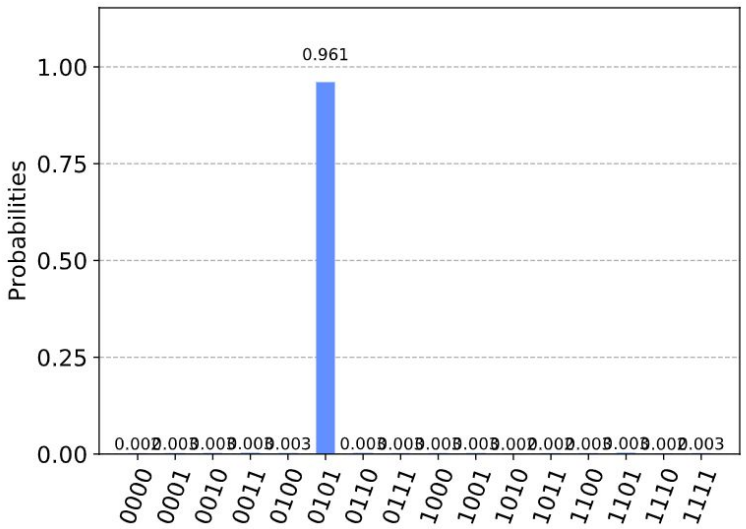


4 Operations

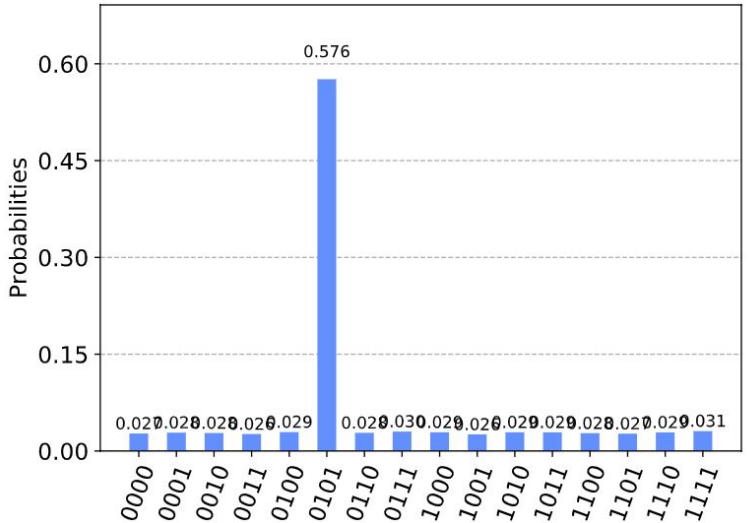


Results continued

6 Operations



8 Operations



Conclusions

- Shared a brief background about Lov Grover
- Explained Grover's algorithm in theory and its benefits
- Demonstrated Grover's Algorithm on 2 qubits to find the state $|00\rangle$
- Simulated Grover's Algorithm on 4 qubits to find the state $|0101\rangle$ on Qiskit
  - After 4 operations, the algorithm finds the state ≈91% of the time
  - The algorithm performs best after 6 operations, finding the state ≈96% of the time

Thank You for Your Attention!

We would like to thank Professor Shabani, Bassel Heiba Elfeky and Billy Strickland for all all their help and support along the way. We would also like to thank our peers in the Quantum Masters group, Elena, Esther, Eva, and Lucy, for being with us in this journey.

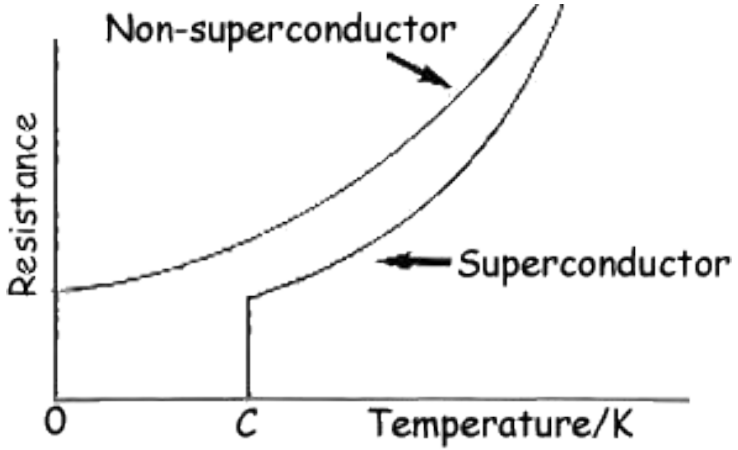
Alex Harris & Ashkan Yazdi Zadeh
Quantum masters
Summer 2020 Qcamp

Elements of Quantum Computing

Ian, Sam, Daniil

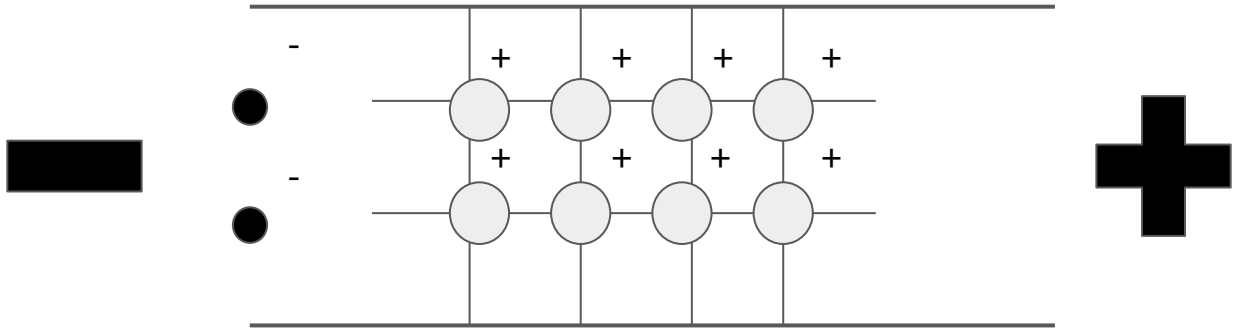
Superconductivity

Superconducting material has R = 0 below a certain (very low) temperature

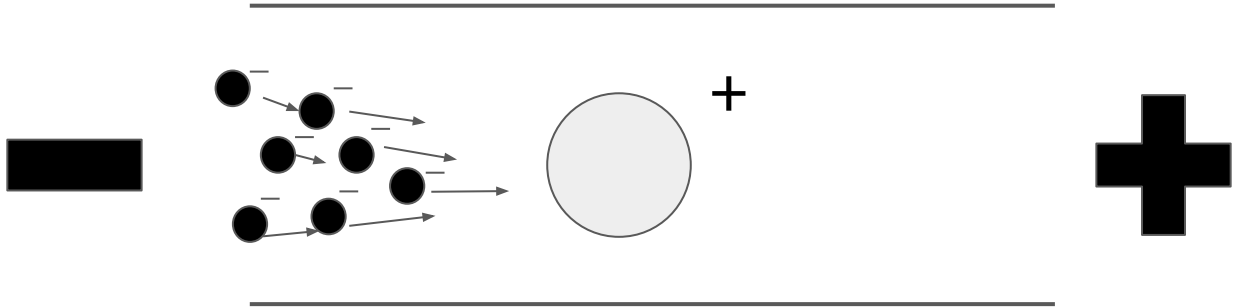


C = critical temperature

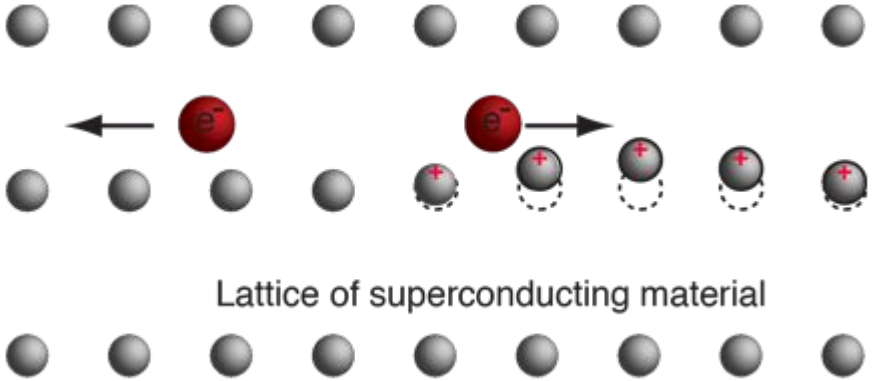
Classical current through lattice of (WLOG) copper atoms



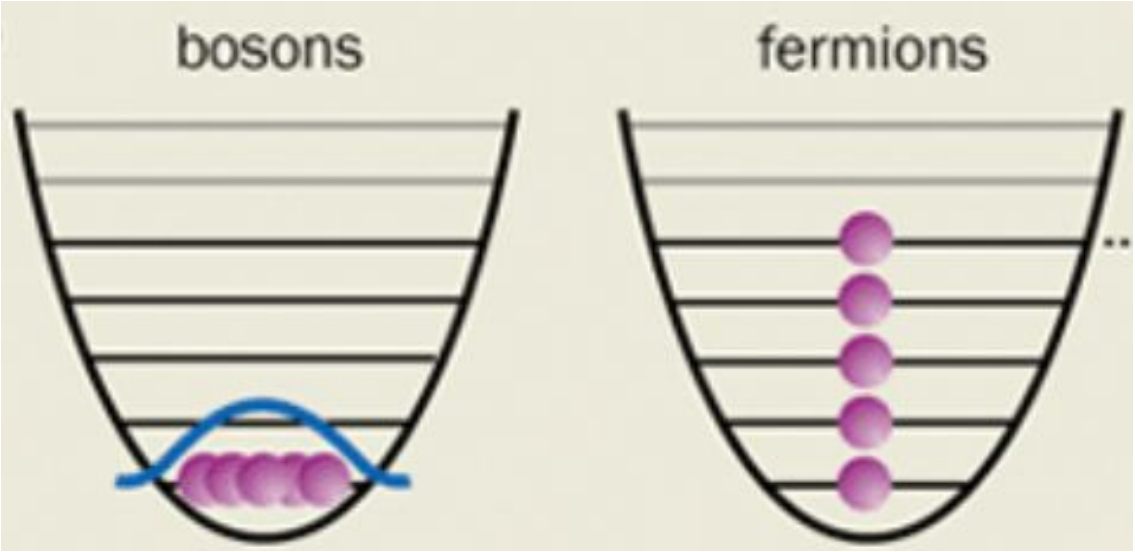
Atoms are slightly distorted towards electrons, forming a centered charge



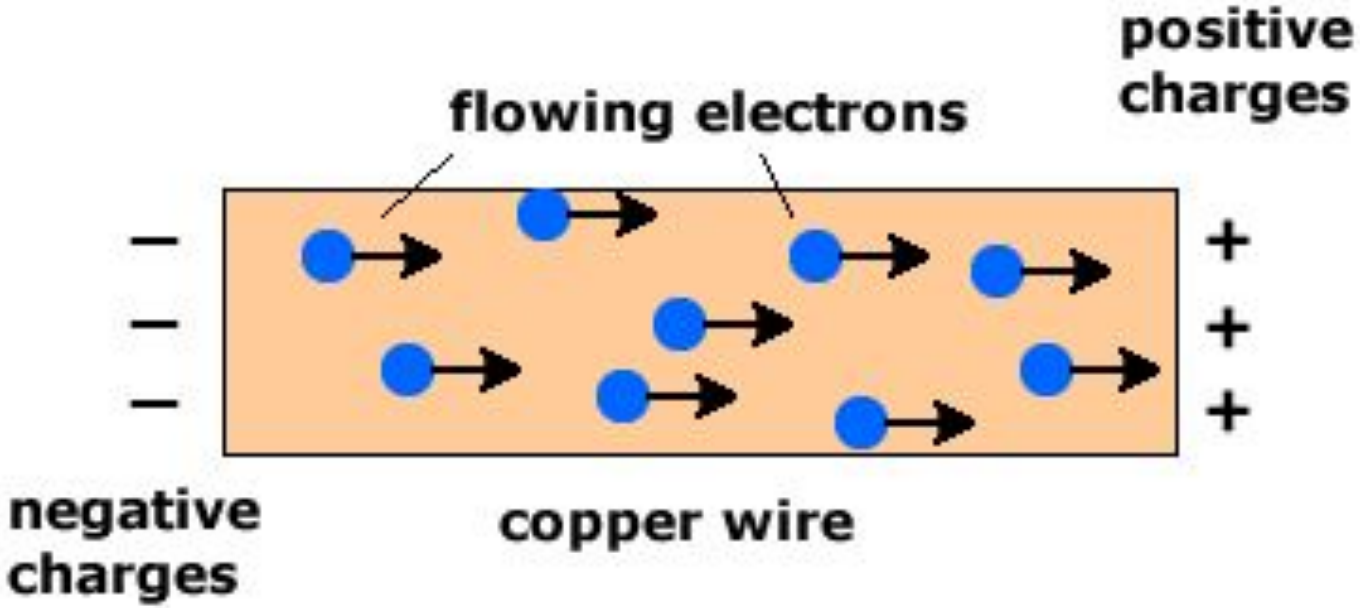
Cooper pairs are formed



Cooper pairs are bosons, and thus not subject to Pauli Exclusion – they can share an energy state



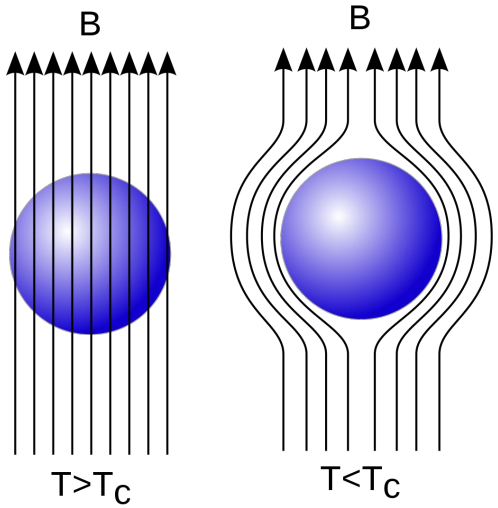
In classical current flow, electrons are “knocked out” when they hit atoms – this is nearly impossible to do with particles in the same energy state



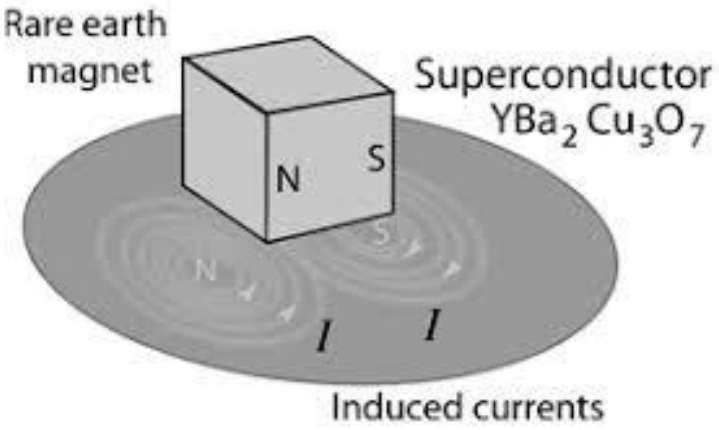
Thus Cooper pairs suffer no resistance from the lattice – no power is lost and current flows infinitely (forever)

$$V = IR$$

An external magnetic field will not flow through a superconducting metal (Meissner Effect)



Surface currents form on superconducting metal, flowing infinitely and generating a magnetic field that pushes out the external field



By considering a superconducting ring in a magnetic field, we can show flux is quantized

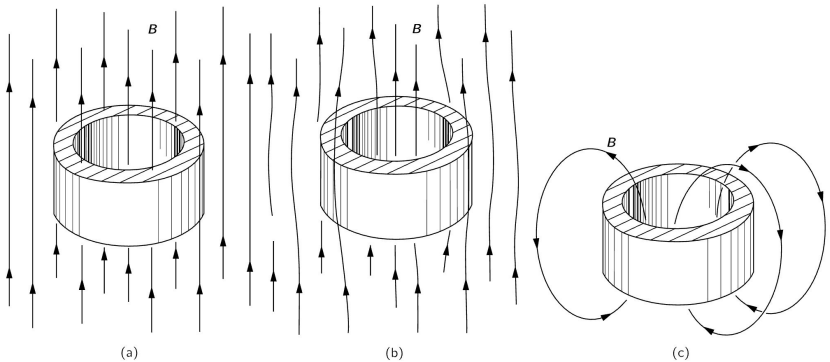


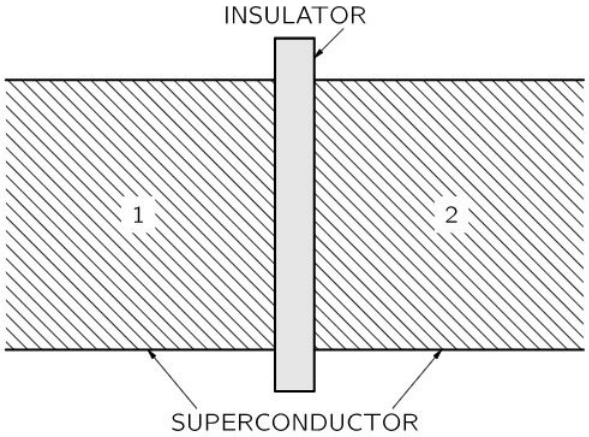
Fig. 21–4. A ring in a magnetic field: (a) in the normal state; (b) in the superconducting state; (c) after the external field is removed.

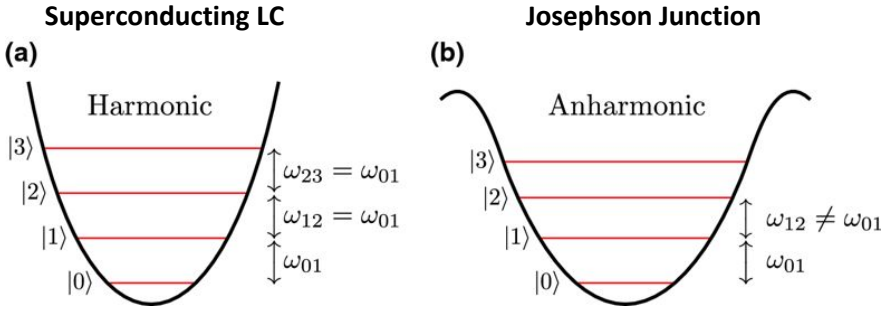
$$2\pi n\hbar = q\Phi$$

Josephson Junctions

-Building block of superconducting qubits

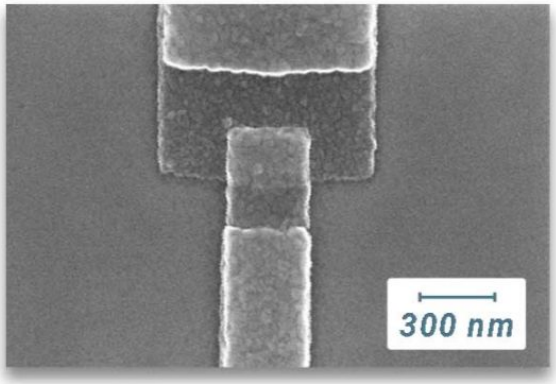
-Usually Al/Al Ox/Al



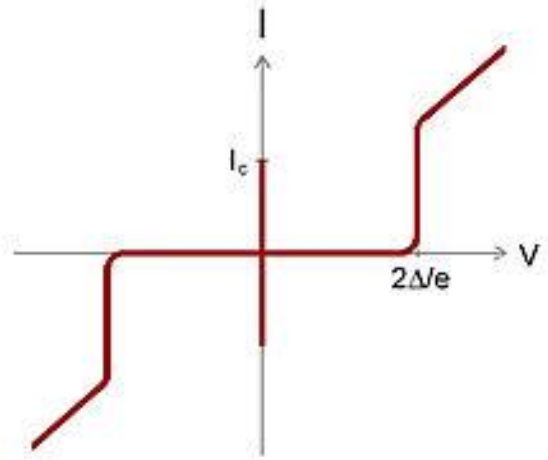


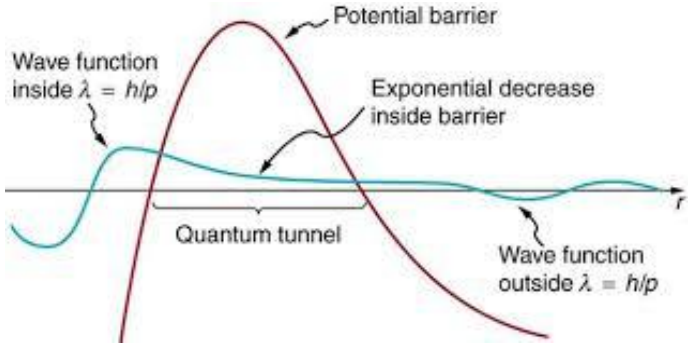
Important Effects

1. **Quantum Tunneling**
2. **The DC Josephson effect**
3. **The AC Josephson effect**

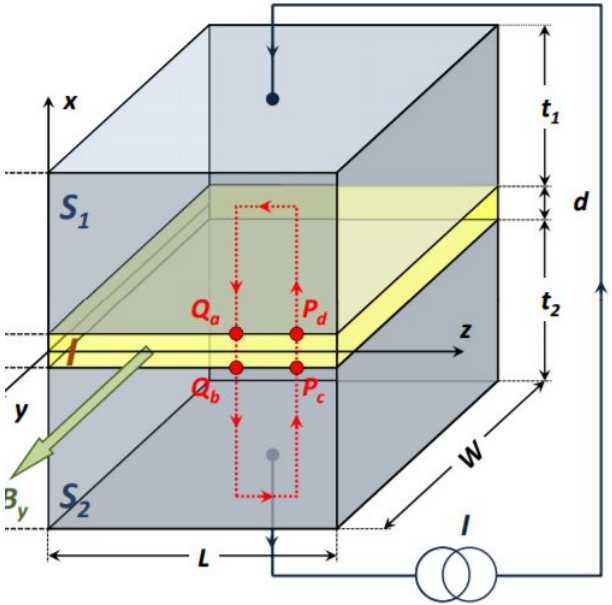


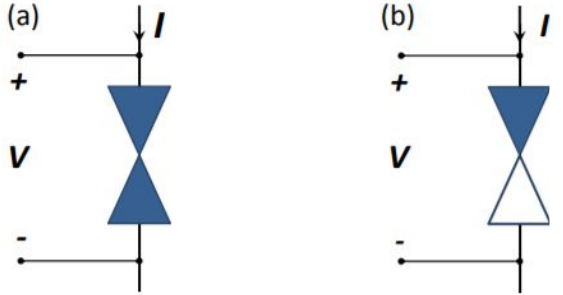
SEM image courtesy of the Institute for Quantum Computing (IQC) at the University of Waterloo





Circuit Models





$$L_s = \frac{\Phi_0}{2\pi I_c \cos\varphi}$$

Finite Voltage Model

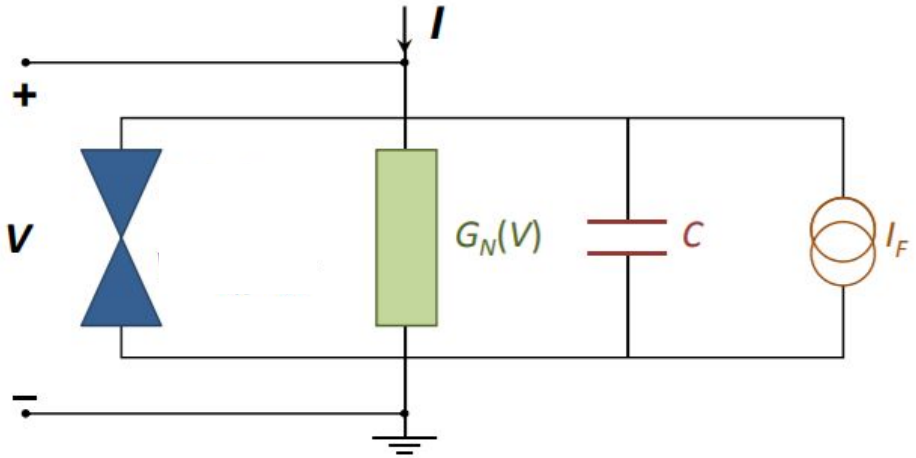
Normal Resistance

$$R_N(V,T) = \begin{cases} \frac{n_{tot}}{n(T)} R_n, |V| < \frac{2\Delta(T)}{e} \\ R_N, |V| > \frac{2\Delta(T)}{e} \end{cases}$$

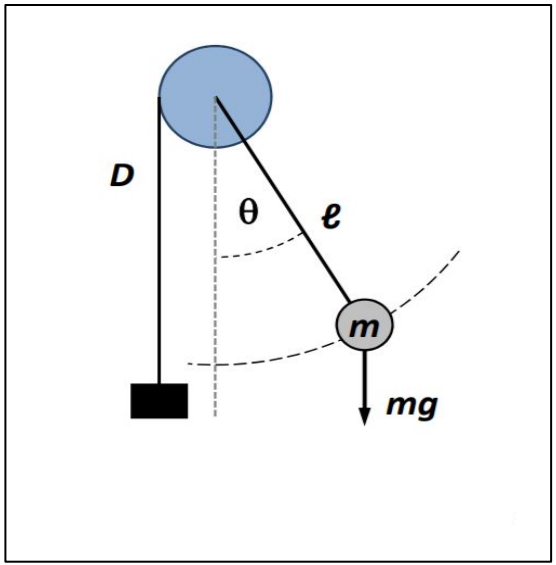
Noise Current (Langevin Method)

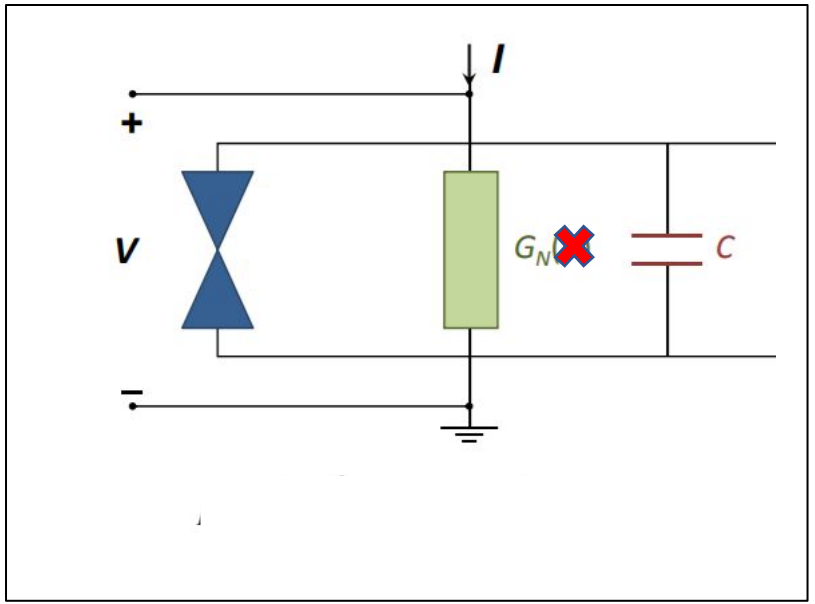
Capacitance

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

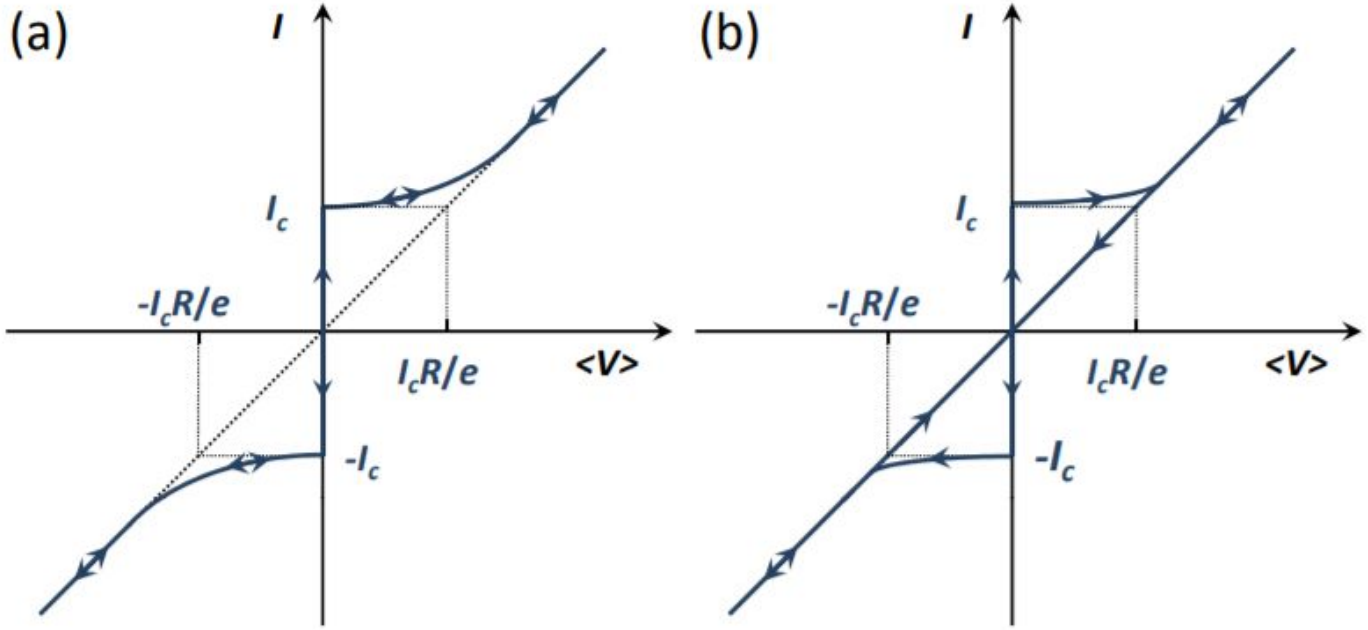


RCSJ Model



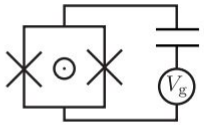
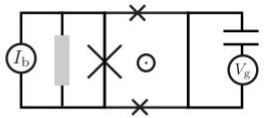
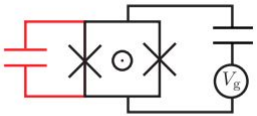
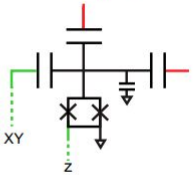
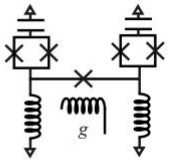
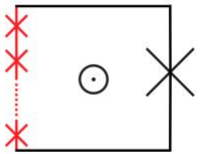
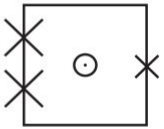
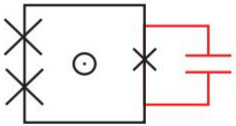
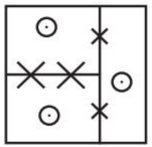


IV Characteristics



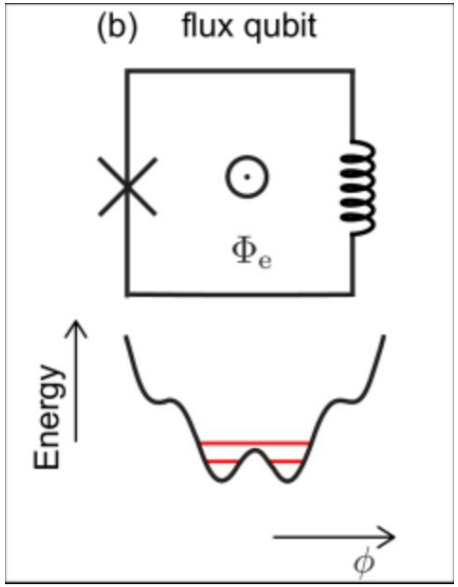
(a) Overdamped (b) Underdamped

Qubits Built from JJ's

Circuit	Split Cooper-pair box 	Quantronium 	Transmon 
Features	Tunable Josephson energy	Charge-flux qubit	Charge noise reduction
Circuit	Xmon 	Gmon 	Fluxonium 
Features	Connectivity	Tunable coupling	Charge noise reduction
Circuit	3-junction flux qubit 	C-shunt flux qubit 	Tunable-gap flux qubit 
Features	Reduction of loop size	Charge noise reduction	Tunability

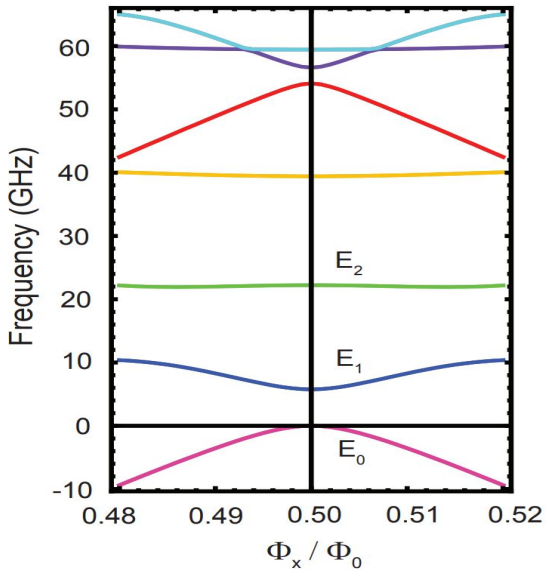
Flux Qubits

$E_J >> E_C$

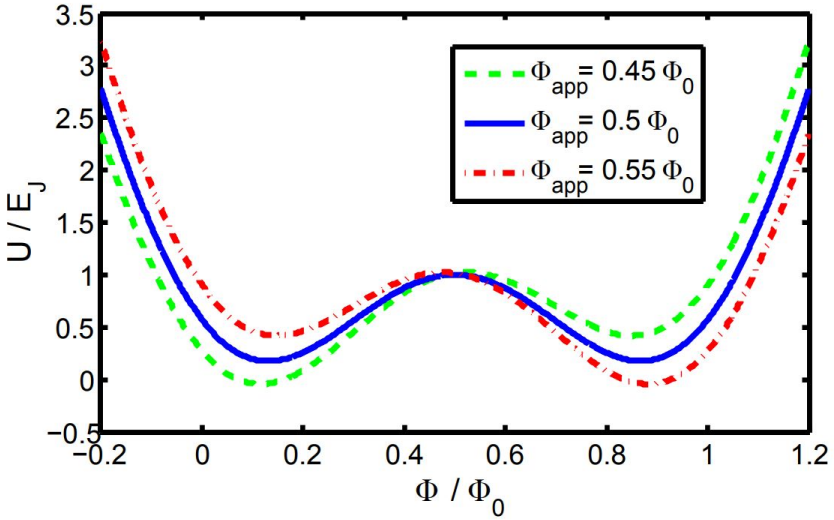


Optimal Working Point

Energy Spectrum vs. Applied Flux

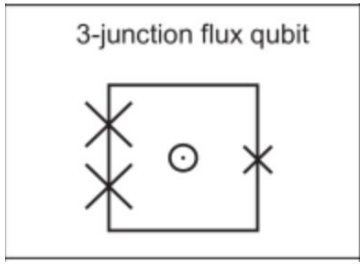


Potential Energy Diagram



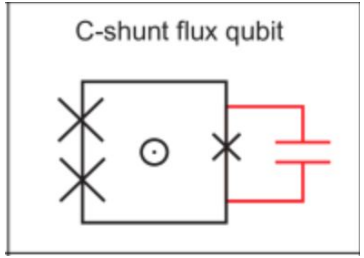
Advantages

- Weak sensitivity to charge noise
- Optimal point

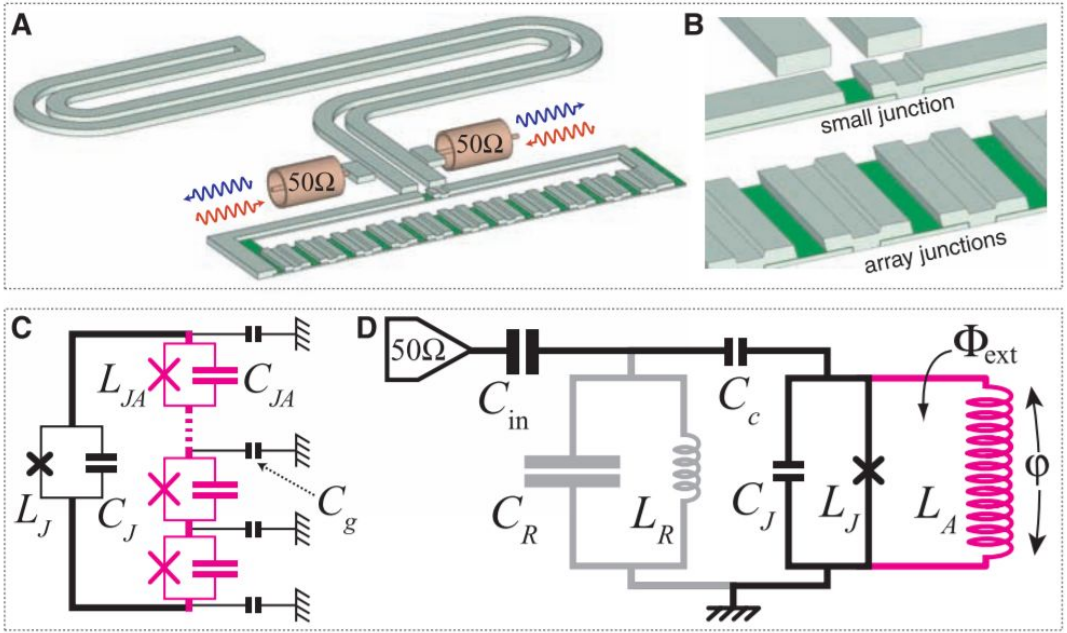


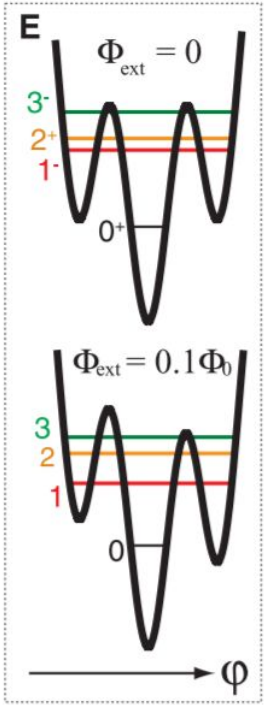
Disadvantages

- Heavy dependence on junction parameters
- Sensitivity to flux noise



Fluxonium Qubits



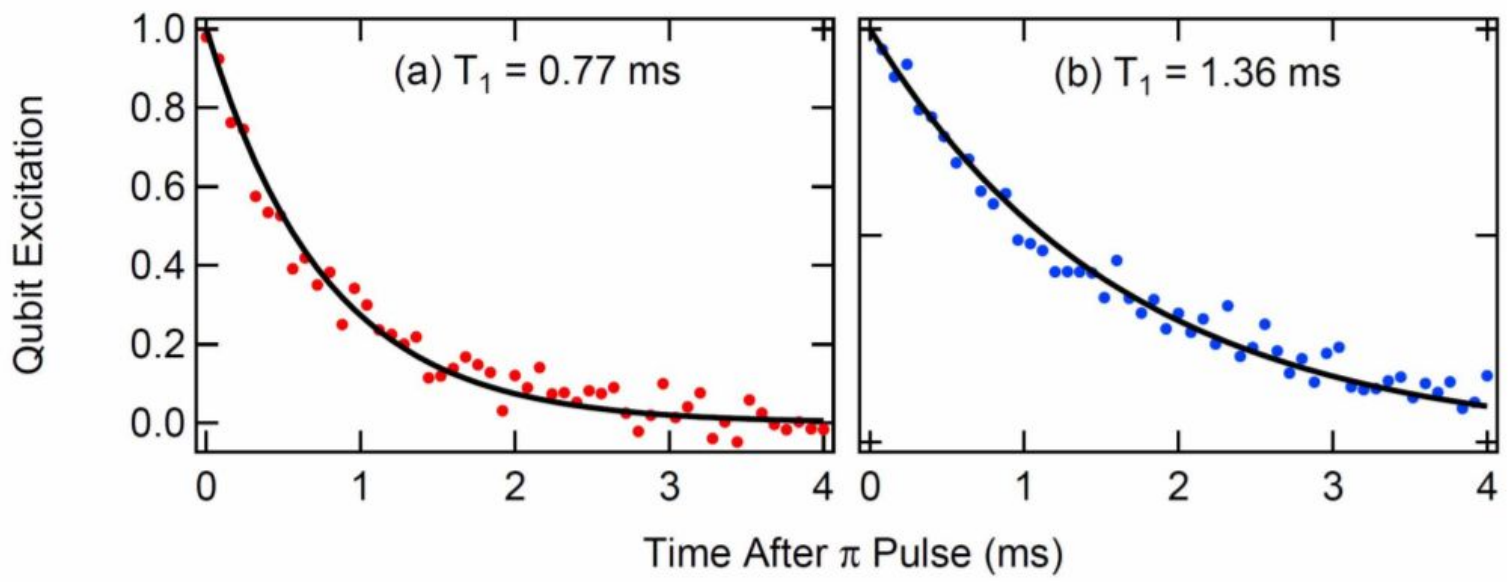


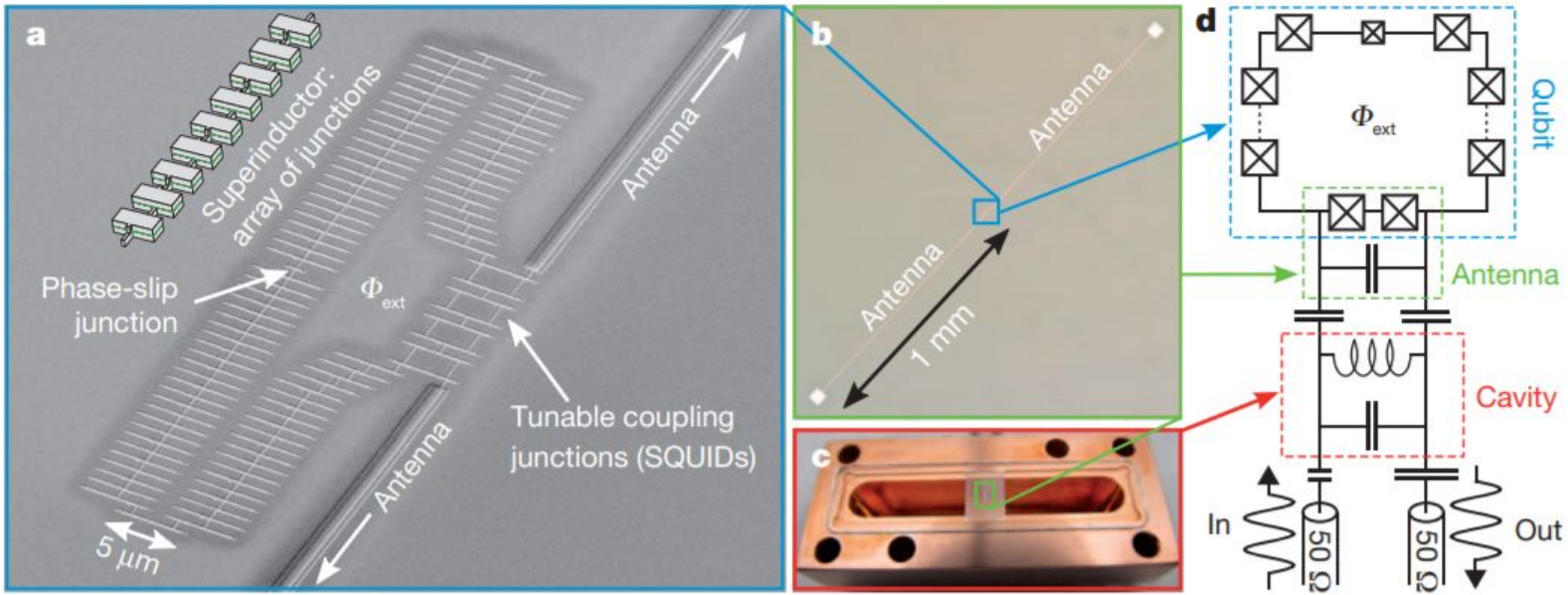
-large inductance, high anharmonicity

-decoherence on order of 100 ns

Next Design (2014)

Max T1 > 1 ms




a
Superinductor:
array of junctions
Antenna
Phase-slip
junction
Φ_{ext}
Tunable coupling
junctions (SQUIDs)
Antenna
5 μm
b
Antenna
Antenna
1 mm
c
d
Φ_{ext}
Qubit
Antenna
Cavity
In
50 Ω
50 Ω
Out

1) Feynman, Richard P. (Richard Phillips), 1918-1988. The Feynman Lectures on Physics.
2) Kockum A.F., Nori F. (2019) Quantum Bits with Josephson Junctions. In: Tafuri F. (eds) Fundamentals and Frontiers of the Josephson Effect. Springer Series in Materials Science, vol 286. Springer, Cham
3) Manucharyan, V. E., Koch, J., Glazman, L. I., & Devoret, M. H. (2009). Fluxonium: Single Cooper-Pair Circuit Free of Charge Offsets. Science, 326(5949), 113–116. doi:10.1126/science.1175552
4) Pop, I. M., Geerlings, K., Catelani, G., Schoelkopf, R. J., Glazman, L. I., & Devoret, M. H. (2014). Coherent suppression of electromagnetic dissipation due to superconducting quasiparticles. Nature, 508(7496), 369–372. doi:10.1038/nature13017
5) Gross, R., & Marx, A. (n.d.). Ch. 2-3. In Applied Superconductivity.
6) Thorough treatment of superconductivity: https://www.feynmanlectures.caltech.edu/III_21.html
7) Intuitive explanation superconductivity: https://www.youtube.com/watch?v=fuloQcljFOs

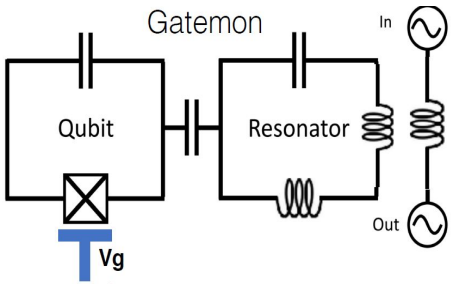
Thank you! Questions?

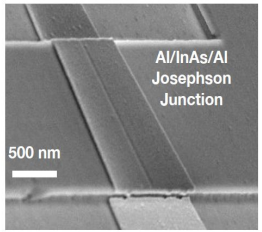
Testing Resonators: Path to Gatemon

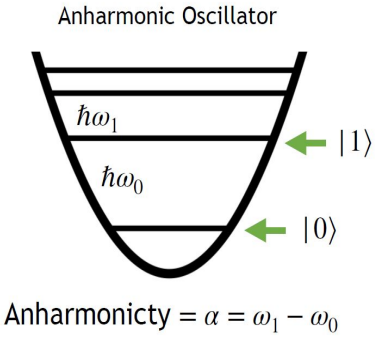
Josh Tong

Image: Google

NYU Gatemon Chip

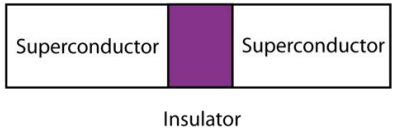


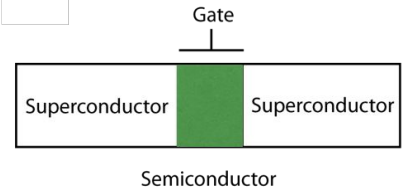




Anharmonicty $= \alpha = \omega_1 - \omega_0$

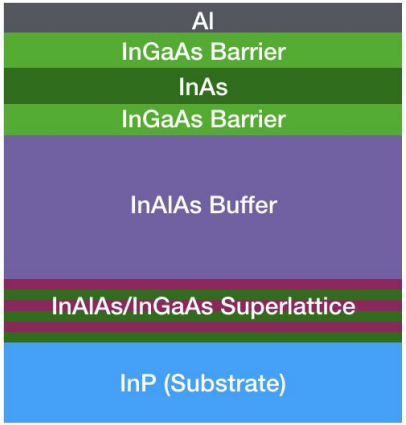
- Semiconductor allows for a frequency tunable qubit with gate voltage!





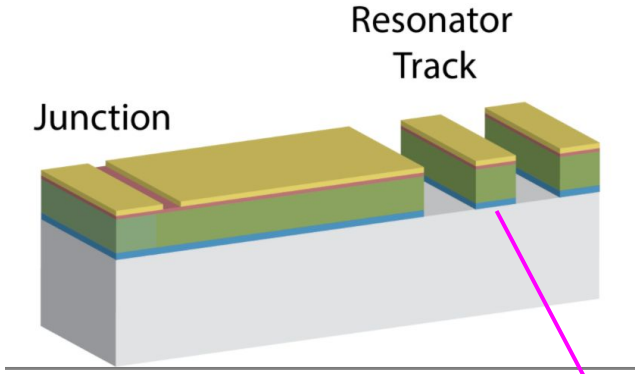
- Good microwave circuitry is essential for the gatemon to work…
- Readout and entangling done through resonator. We want to test how good it is... But, what is it?

Resonator

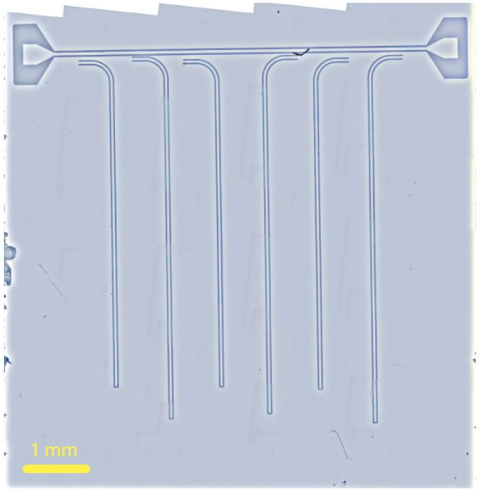


(1) growth

(2) etching



(3) geometry

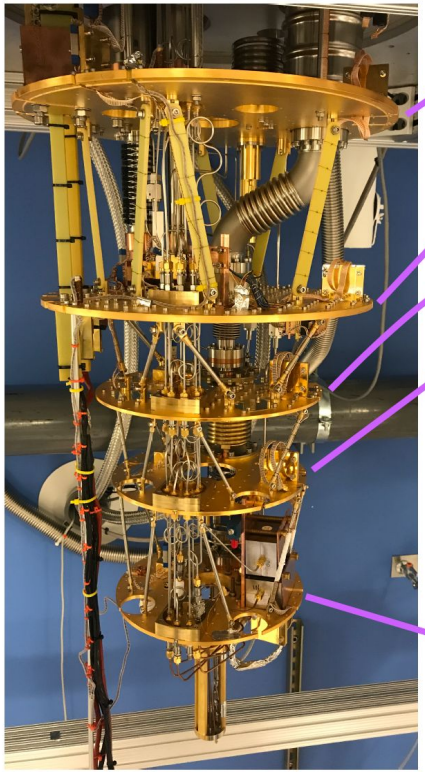


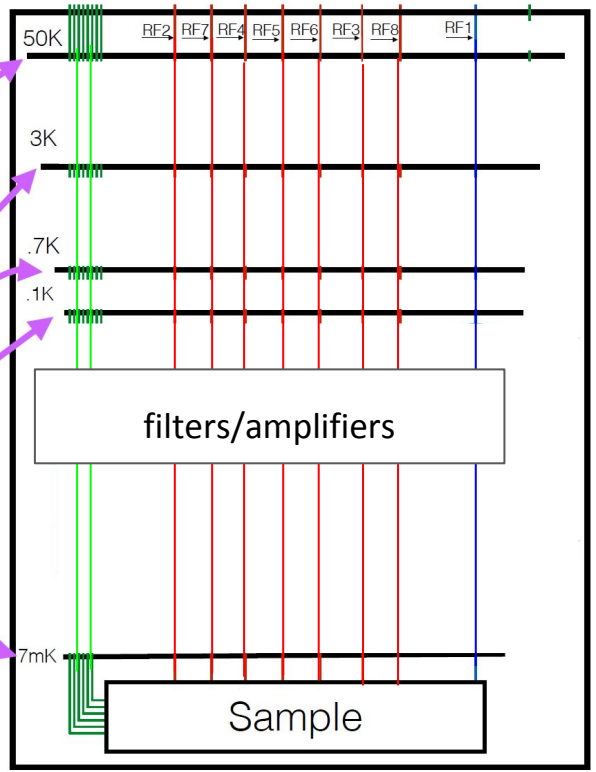
Coplanar waveguide resonates at specific frequency

$$f_0 = v_0/4d$$

Resonates like an instrument!

Measurement

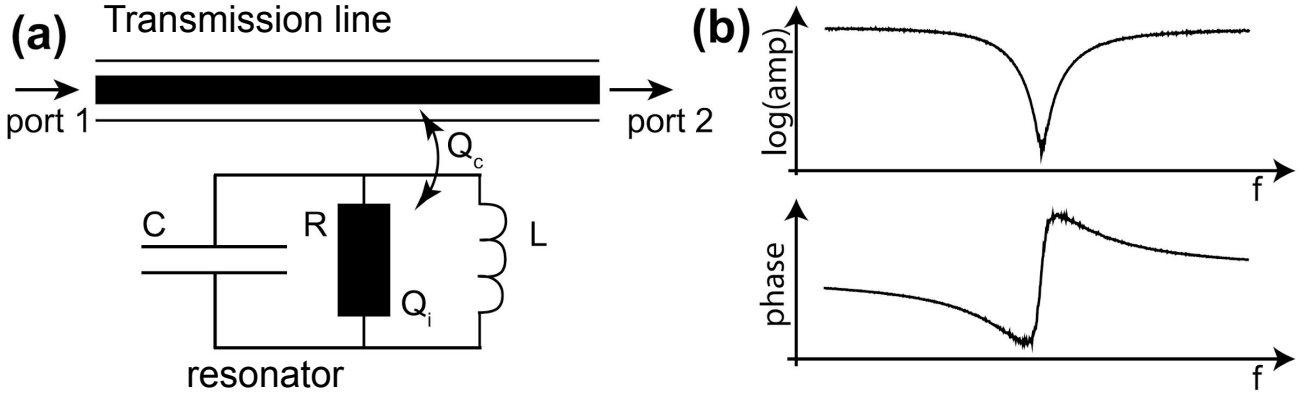




Receive a complex valued signal! (real and imaginary)

Credit: Joe

Transmission Model



Model:

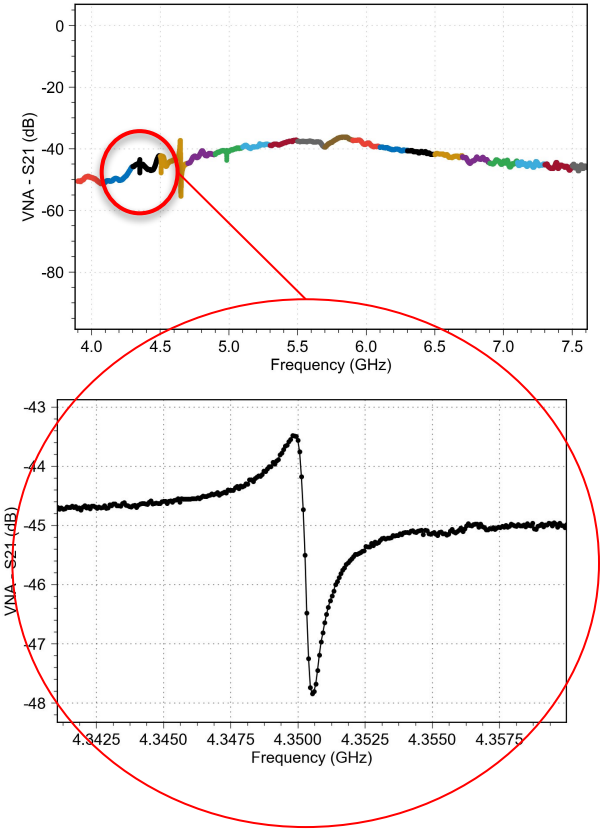
$$S_{21}^{\text{notch}}(f) = \underbrace{ae^{i\alpha}e^{-2\pi i f\tau}}_{\text{environment}}\underbrace{\left[1 - \frac{(Q_l/|Q_c|)\ e^{i\phi}}{1 + 2iQ_l\,(f/f_r - 1)}\right]}_{\text{ideal resonator}}$$

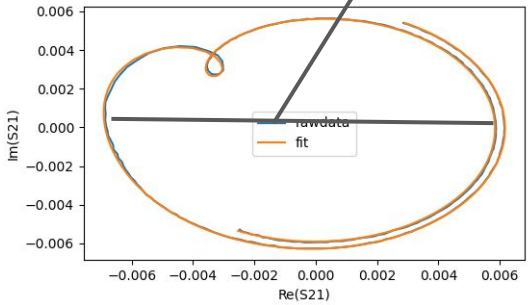
$$Q_l^{-1} = Q_i^{-1} + \text{Re}\left\{Q_c^{-1}\right\}$$

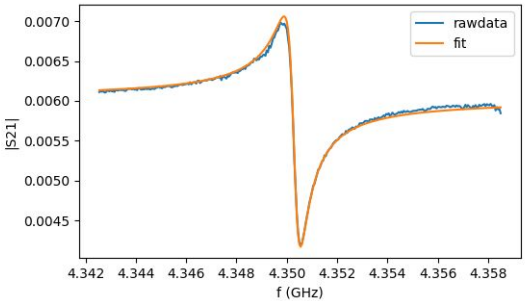
a is an additional amplitude, α is due to phase shift, τ electronic delay due to cable length, ϕ is due to impedance mismatch

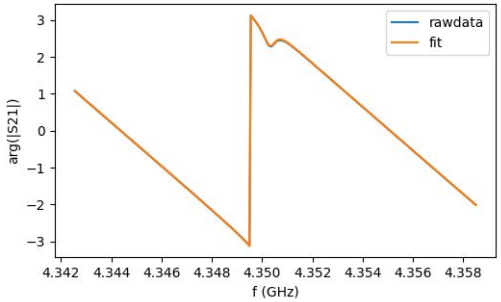
Scan Procedure

$$d = Q_l / |Q_c|$$



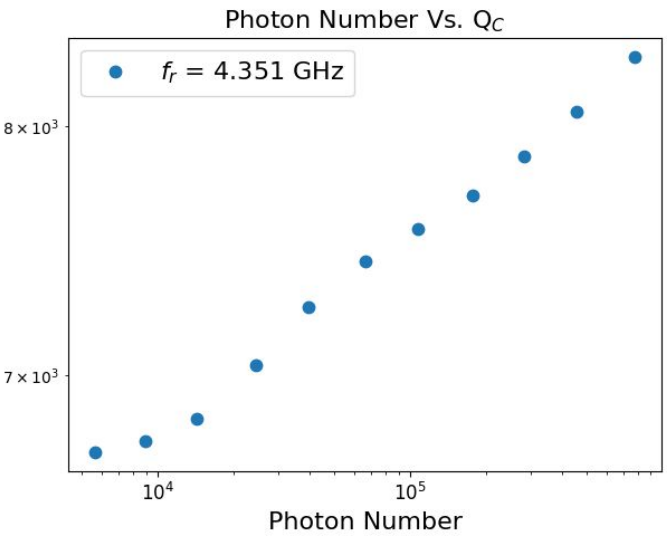


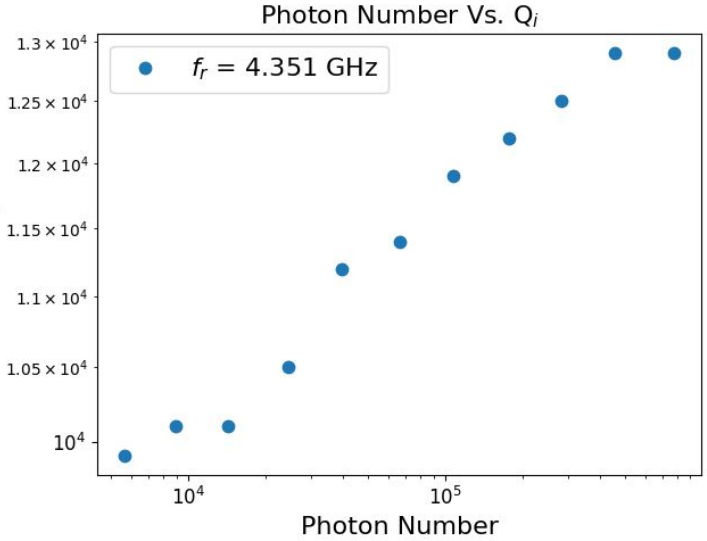




$$Q_l^{-1} = Q_i^{-1} + \mathrm{Re}\left\{Q_c^{-1}\right\}$$

Power vs Q

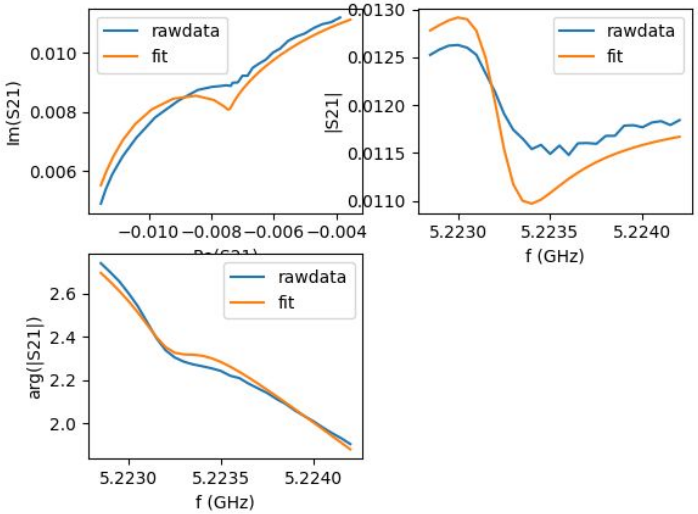




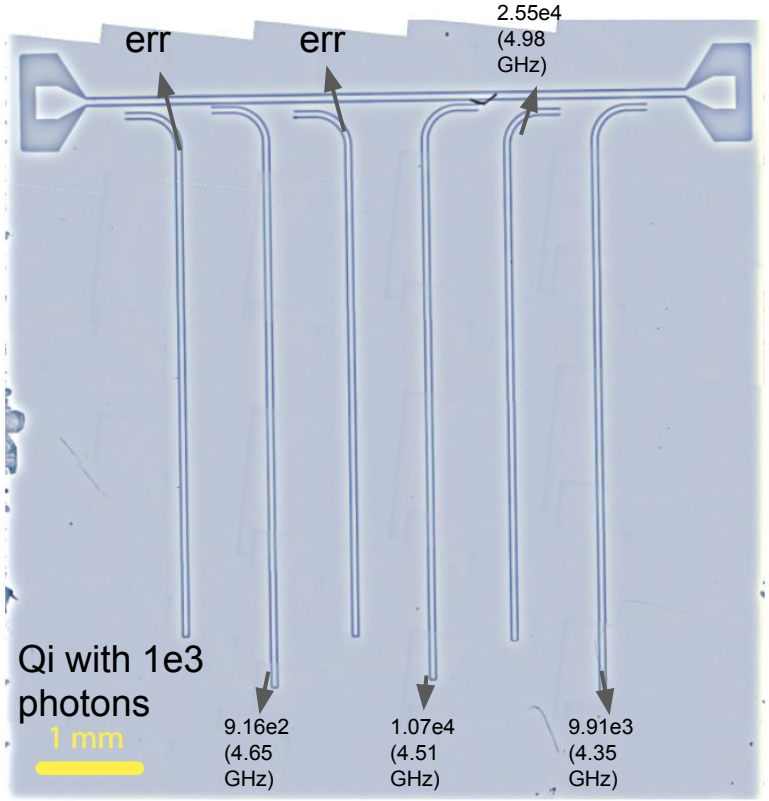
- Q gets **better** with increasing power because imperfections (2 level systems) that absorb photons are **saturated**
- We want to operate at low power (ideally single photon)
- We care about Qi because this tells us how good the chip is

Q for each resonator

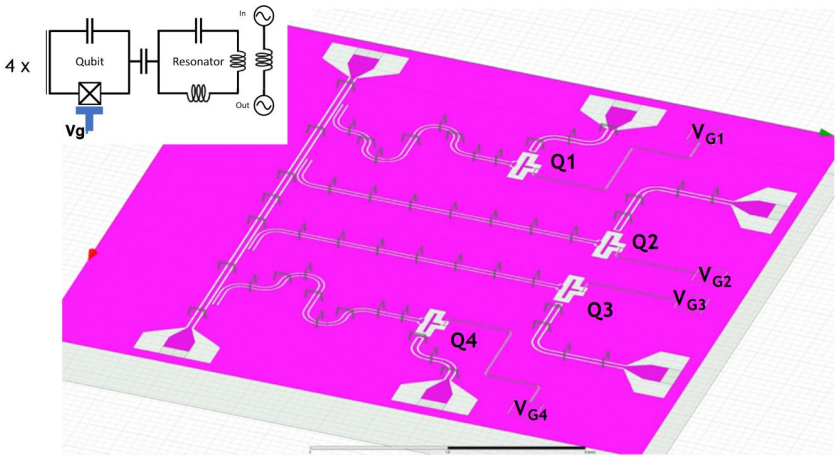
Hard to fit….

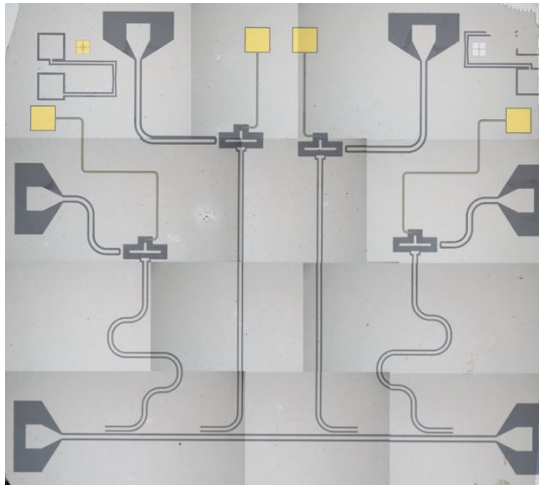


Work to be done to improve fitting algorithm and growth



When we do that we can build gatemon chip!





Thank you, questions?

Thanks to wizards, Bassel, Joe, Billy, Prof. Shabani

Qiskit

Option Pricing with Qiskit

Mark Koszykowski

Shabani Lab

Intro to Options

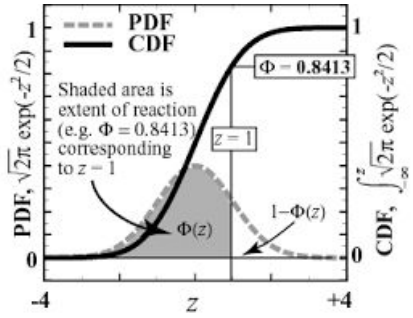
2 kinds of Option Contracts:

- Call Options
- Put Options

3 kinds of Options:

- European Options
- American Options
- Bermudan Options

Pricing Options (Black Scholes)



- Stochastic Differential Equation: $\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t$
- Formula for European Call Options: $C^{eu} = S_0\Phi(d_1) - K\mathrm{e}^{-rT}\Phi(d_2)$
- Formula for European Put Options: $P^{eu} = K\mathrm{e}^{-rT}\Phi(-d_2) - S_0\Phi(-d_1)$

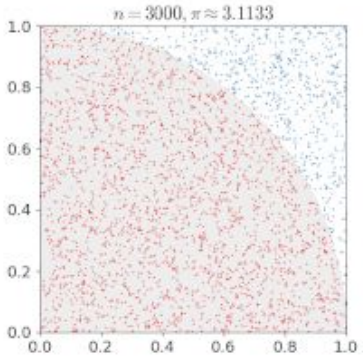
Where S is the price of the stock, K is the exercise price, r is the risk-free interest, T is the time to expiration, σ is the volatility, and Φ is a CDF for Standard Distribution

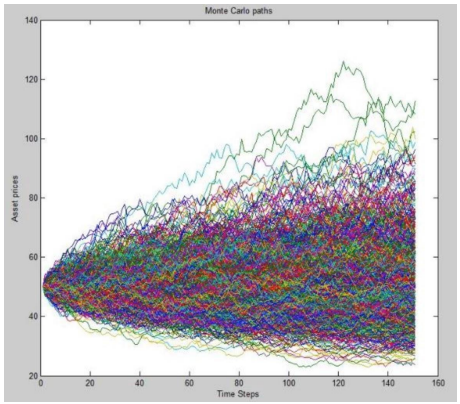
$$d_1 = \frac{\log\left(\frac{S_0}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

Monte Carlo Methods

- Broad class of algorithms that utilizes randomness and expectation to make predictions
- Variety of applications but are often used in finance to predict the value or fair price of an option



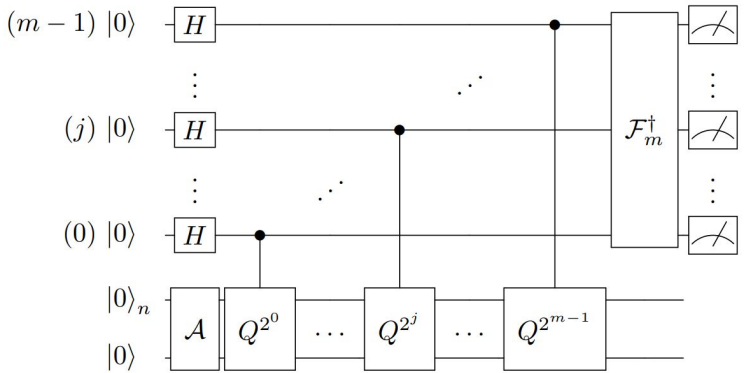


Amplitude Estimation

- Quantum algorithm that is used to replace Quantum Monte Marlo Methods
- Combines ideas from Grover's Algorithm and Shor's Algorithm

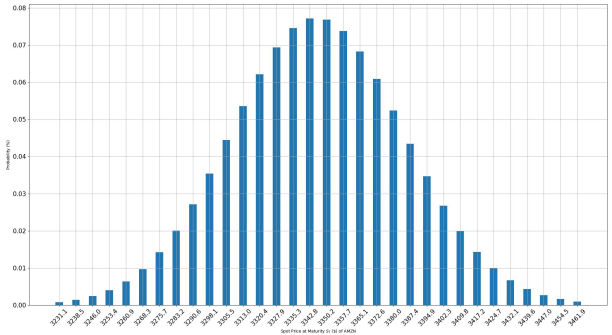
$$\mathcal{A}\left|0\right\rangle_{n+1} = \sqrt{1-a}\left|\psi_0\right\rangle_n\left|0\right\rangle + \sqrt{a}\left|\psi_1\right\rangle_n\left|1\right\rangle$$

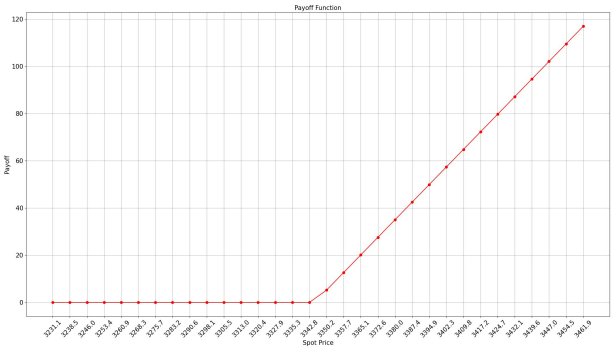
$$\mathcal{Q} = \mathcal{A}\mathcal{S}_0\mathcal{A}^\dagger\mathcal{S}_{\psi_0}$$



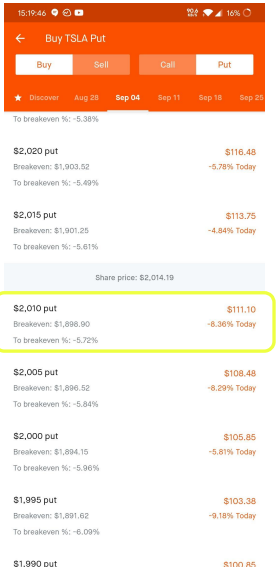
My Final Project Pseudocode

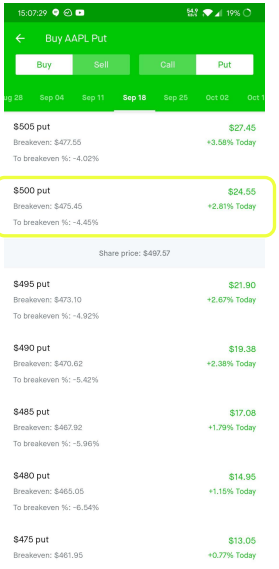
- Receive necessary information from user about option
- Calculate the derived stock's volatility (SDLR)
- Produce a Log Normal Uncertainty Model
- Determine the Payoff for each probabilistic price
- Feed information into Amplitude Estimation algorithm
- Output the estimated payoff

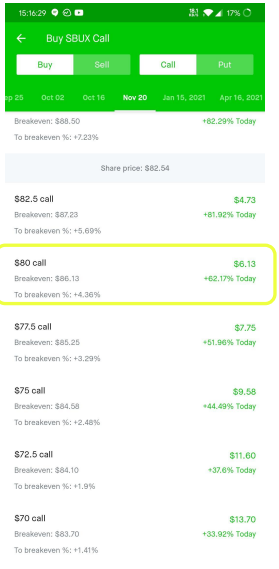


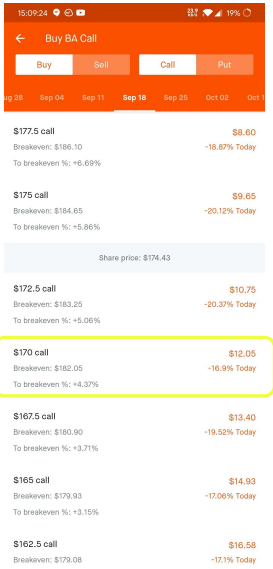


Results










```
Please enter the name of the stock which the option is derived from (ex. AAPL): TSLA
Please enter the strike price of the option (ex. 1500.00): 2010
Please enter the annual interest rate as a percentage (ex. 5; default is 0%):
Please enter the date at which the option expires (ex. 2020-08-25): 2020-09-04
Please enter whether this option is a Call or Put (ex. C or P): P
Estimated payoff:       73.1686
Confidence interval:    [51.3371, 94.9842]
```

```
Please enter the name of the stock which the option is derived from (ex. AAPL): AAPL
Please enter the strike price of the option (ex. 1500.00): 500
Please enter the annual interest rate as a percentage (ex. 5; default is 0%):
Please enter the date at which the option expires (ex. 2020-08-25): 2020-09-18
Please enter whether this option is a Call or Put (ex. C or P): P
Estimated payoff:       20.9978
Confidence interval:    [14.9740, 27.0280]
```

```
Please enter the name of the stock which the option is derived from (ex. AAPL): SBUX
Please enter the strike price of the option (ex. 1500.00): 80
Please enter the annual interest rate as a percentage (ex. 5; default is 0%):
Please enter the date at which the option expires (ex. 2020-08-25): 2020-11-20
Please enter whether this option is a Call or Put (ex. C or P): C
Estimated payoff:       5.2734
Confidence interval:    [3.1957, 7.3512]
```

```
Please enter the name of the stock which the option is derived from (ex. AAPL): BA
Please enter the strike price of the option (ex. 1500.00): 170
Please enter the annual interest rate as a percentage (ex. 5; default is 0%):
Please enter the date at which the option expires (ex. 2020-08-25): 2020-09-18
Please enter whether this option is a Call or Put (ex. C or P): C
Estimated payoff:       10.0917
Confidence interval:    [7.8848, 12.3785]
```

Future Improvements

- Including more variables of uncertainty than just volatility
- Calculating value of options whose strike prices are almost 0%
- Trying to create Option Pricer with Quantum Machine Learning (qGAN)

References

- https://arxiv.org/pdf/1905.02666.pdf
- https://arxiv.org/pdf/1806.06893.pdf
- https://arxiv.org/pdf/quant-ph/0005055.pdf
- https://qiskit.org/documentation/tutorials/finance/04_european_put_option_pricing.html
- https://qiskit.org/documentation/tutorials/finance/03_european_call_option_pricing.html
- https://www.cmi.ac.in/~shariq/Shariq%20files/option_pricing.pdf

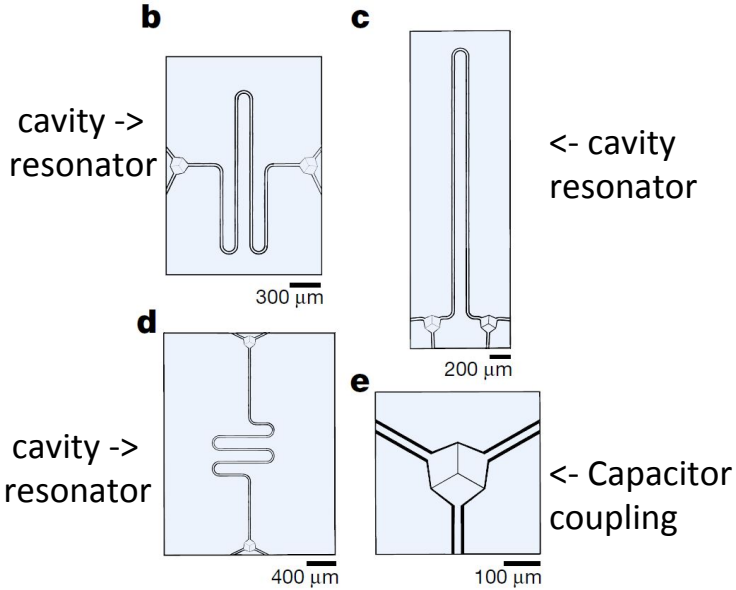
Circuit QED Lattices

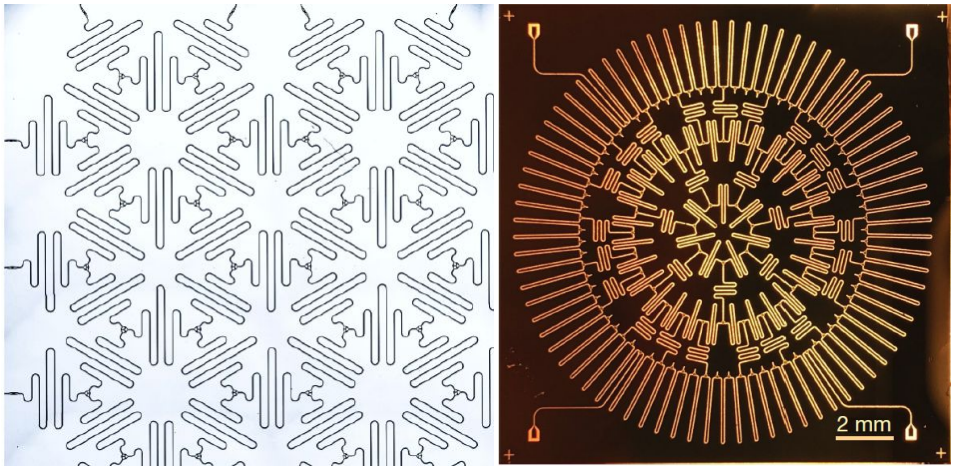
Dylan Langone

Quantum Materials and Devices Lab: New York University

What are we working with?

- Superconducting qubits coupled to networks of microwave cavity resonators

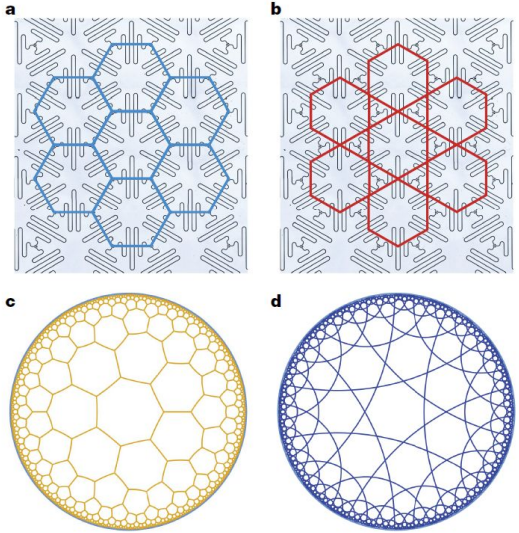




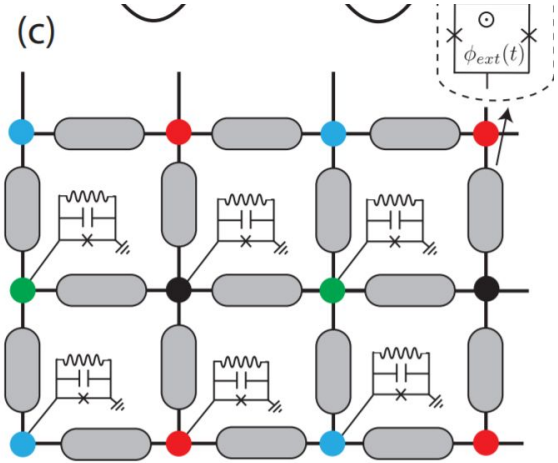
Figures from

Why are we working with them?

- Simulates physics of materials Euclidean and hyperbolic space
- Construction of Hamiltonians of systems exhibiting topological ordering i.e. fractional quantum Hall [2-4]



Lattice for simulating Euclidean materials (a, b) and hyperbolic (c, d). Used from [1]



Network of anharmonic cavity resonators coupled to superconducting qubits simulates a Hamiltonian with interesting topological properties. Used from [2].

Physics of Tight-binding Resonators

- Hamiltonian is the sum over resonators of each on-site energy and interaction with nearest neighbors

$$\mathcal{H}_{eff} = \sum_i \omega_0 a_i^\dagger a_i - \sum_{\langle i,j \rangle} t_{i,j}(a_i^\dagger a_j + a_j^\dagger a_i),$$

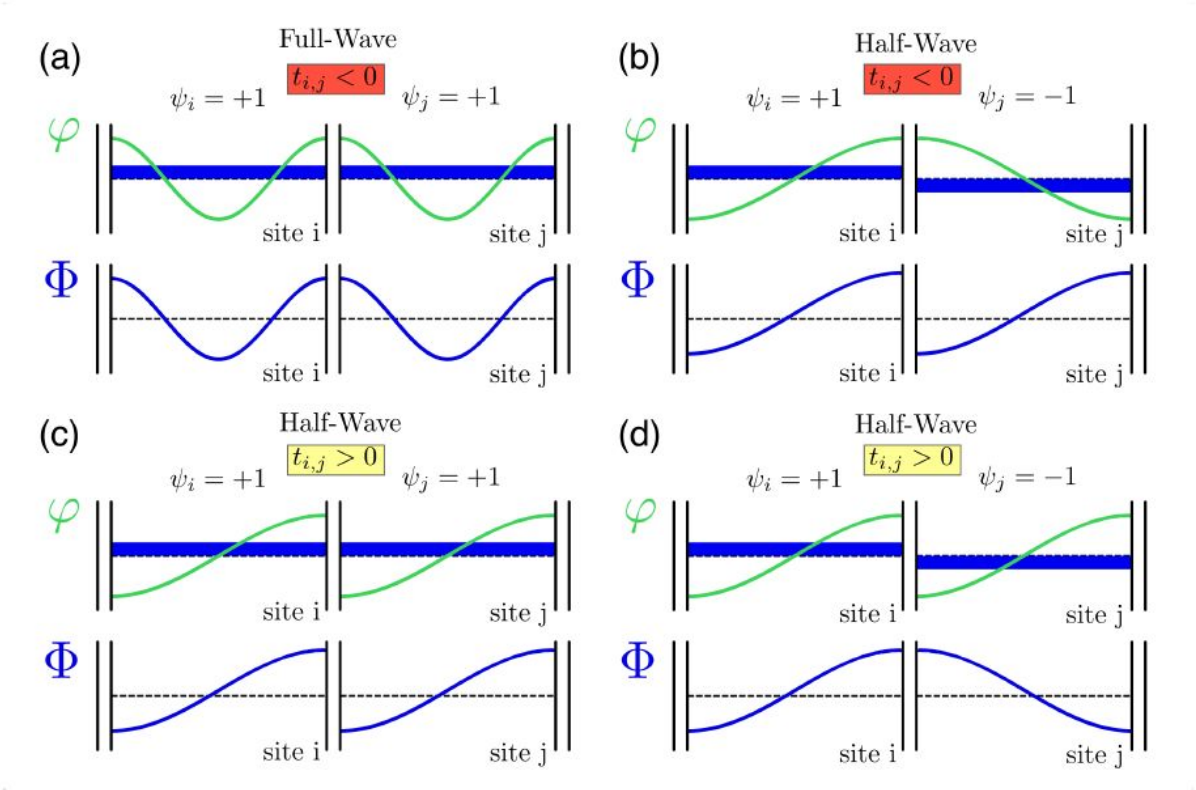
$$H_{eff} = -\sum_{\langle i,j \rangle} t_{i,j}(a_i^\dagger a_j + a_j^\dagger a_i),$$

Hopping matrix element -> determines strength of interaction between nearest-neighbor resonators

$$t_{i,j} = -\frac{1}{2}\omega_0 C_c \times \varphi(i, x_{end}^{(i)}) \times \varphi(j, x_{end}^{(j)})$$

Resonator Wavefunctions

We are only using the
full-wave mode



Figures from [1]

QED lattice graph Hamiltonian

Let $w_s : \mathcal{E} \times \mathcal{E} \to \{0, 1\}$ be given by

$$w_s(e, e') = \begin{cases} 1, & \text{if } e \text{ and } e' \text{ share a vertex} \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

Given an orientation of the edges of X denote by e^+ and e^- the head and foot in V of an edge e. Define $w_a : \mathcal{E} \times \mathcal{E} \to \{0, 1, -1\}$ by

$$w_a(f, g) = \begin{cases} 1, & \text{if } f^+ = g^+ \text{ or } f^- = g^- \\ -1 & \text{if } f^+ = g^- \text{ or } f^- = g^+, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Note that $w_\alpha(e, e')$ is symmetric in e, e' for $\alpha = a$ or $\alpha = s$.

The vector space of functions $f : \mathcal{E} \to \mathbb{C}$ comes with an inner product

$$\langle f, g \rangle = \sum_{e \in \mathcal{E}} f(e) g^*(e), \tag{20}$$

and we denote the inner product space by $\ell^2(\mathcal{E})$. The effective tight-binding Hamiltonians $\bar{H}_\alpha(X)$ for $\alpha = s$ or a on $\ell^2(\mathcal{E})$ that were introduced in Sec. III are given in terms of w_α by

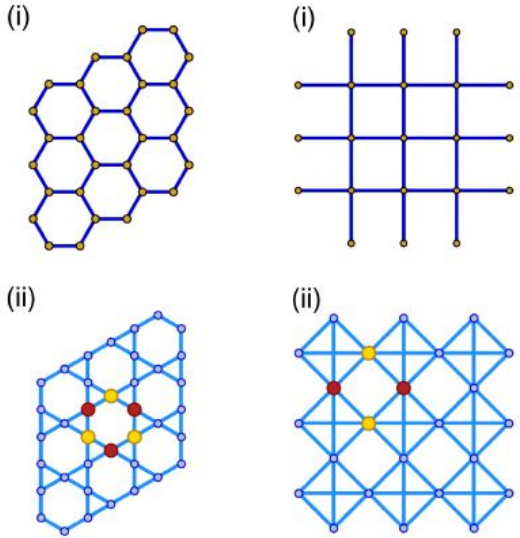
$$\bar{H}_\alpha f(e) = \sum_{e' \in \mathcal{E}} w_\alpha(e, e') f(e'). \tag{21}$$

Hamiltonians are adjacency matrices for full-wave modes

QED lattice graph Hamiltonian

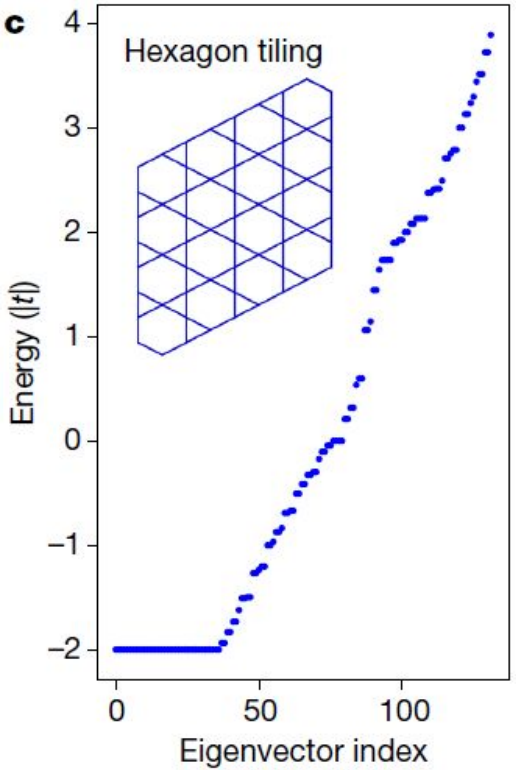
- Hamiltonian is the adjacency matrix of the line graph of the lattice layout:

$$\bar{H}_s(X) = H_{L(X)} = -tA_{L(X)}$$

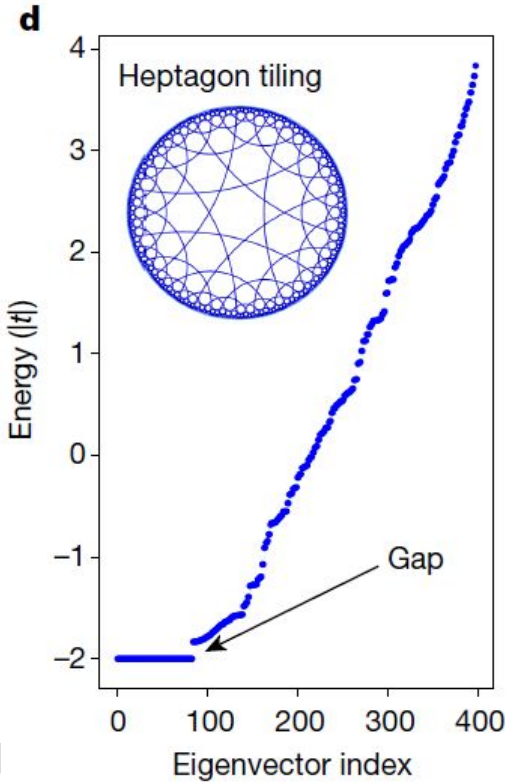


Figures from [1]

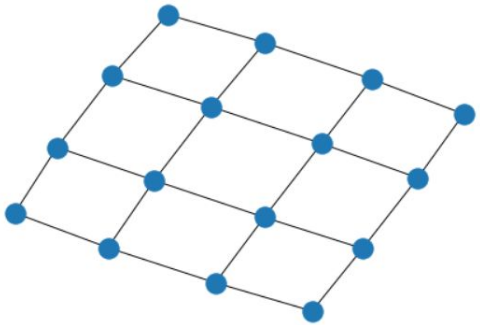
Spectra Examples

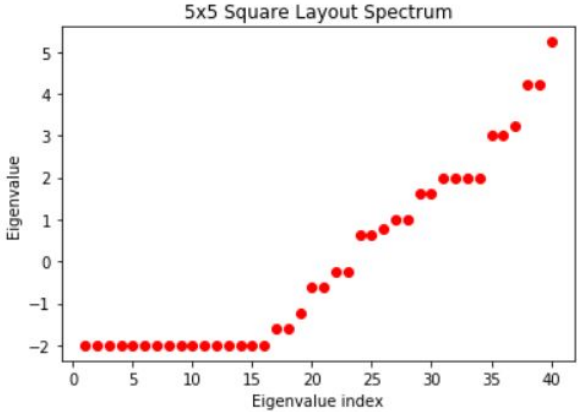


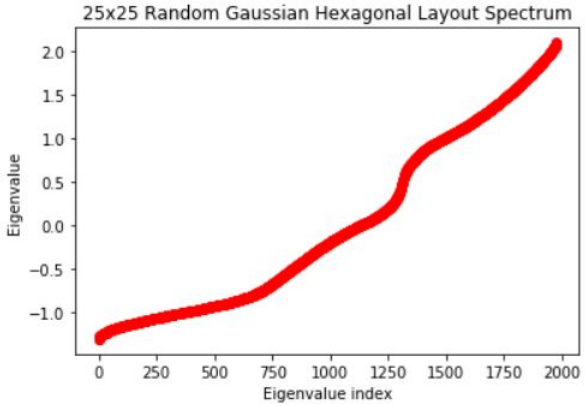
Figures from [1]



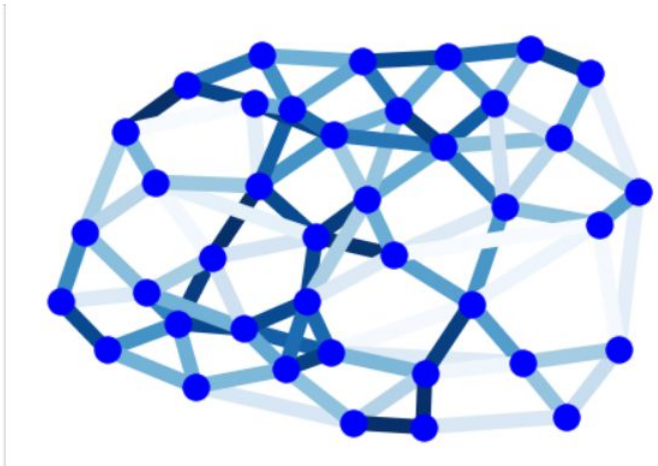
Verification of Spectra

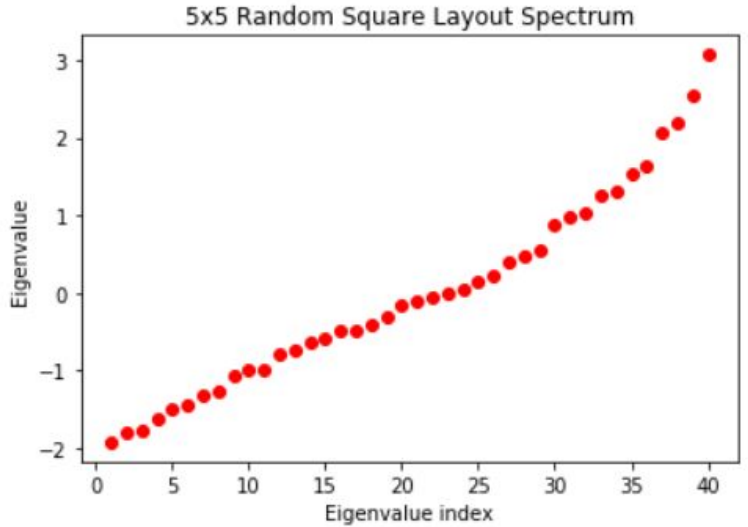






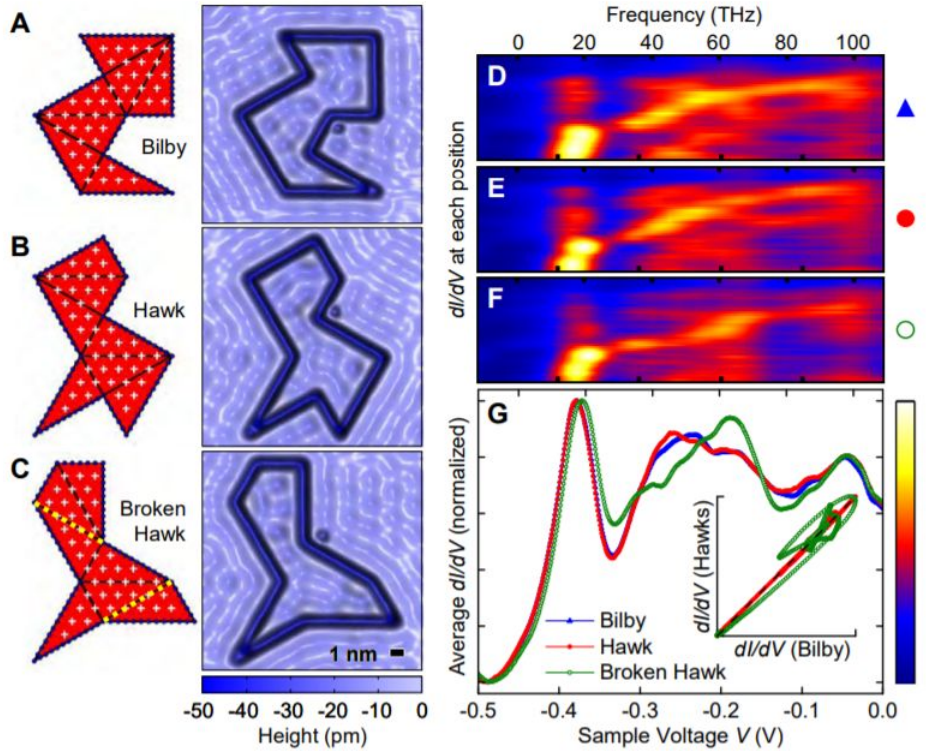
Random Spectra





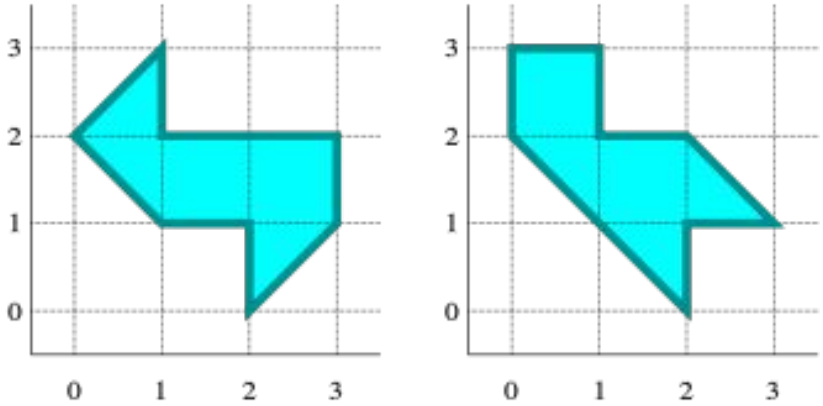
Isospectral Layouts

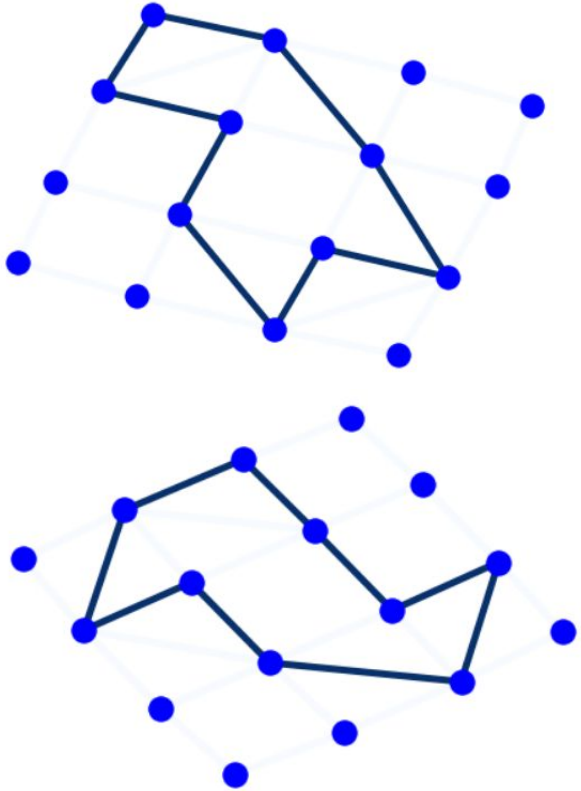
- Experiment [5] demonstrated how to extract quantum phase from isospectral layouts by exploiting topological degree of freedom introduced by isospectrality
- Do energy levels cross in a transition between isospectral configurations?



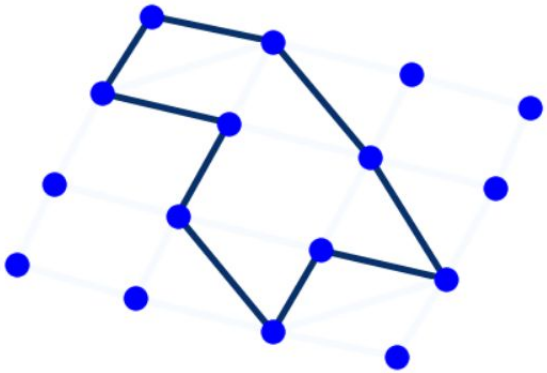
Figures from [5]

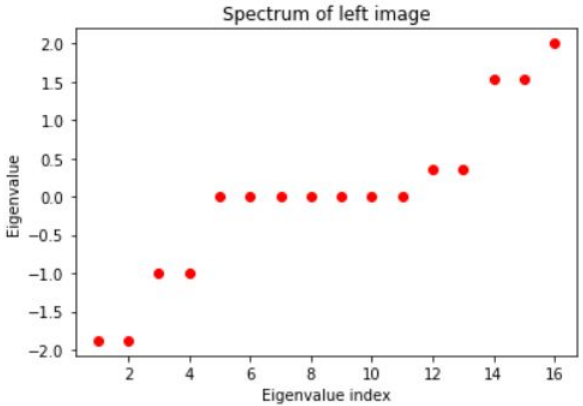
Isospectral Configuration Transition

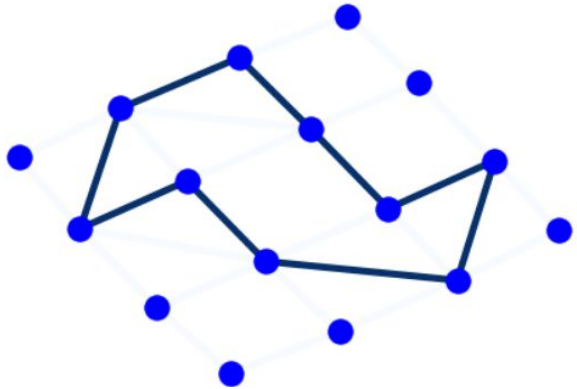


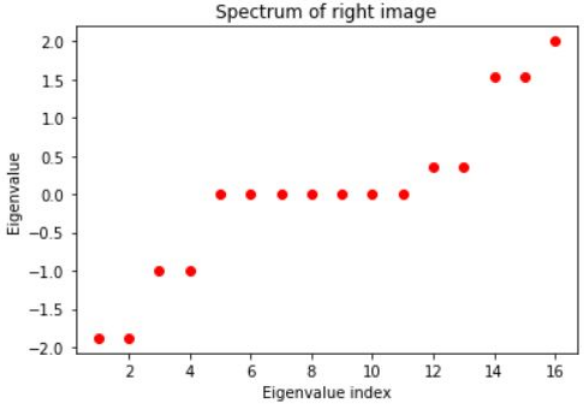


Verification of Isospectral Behavior

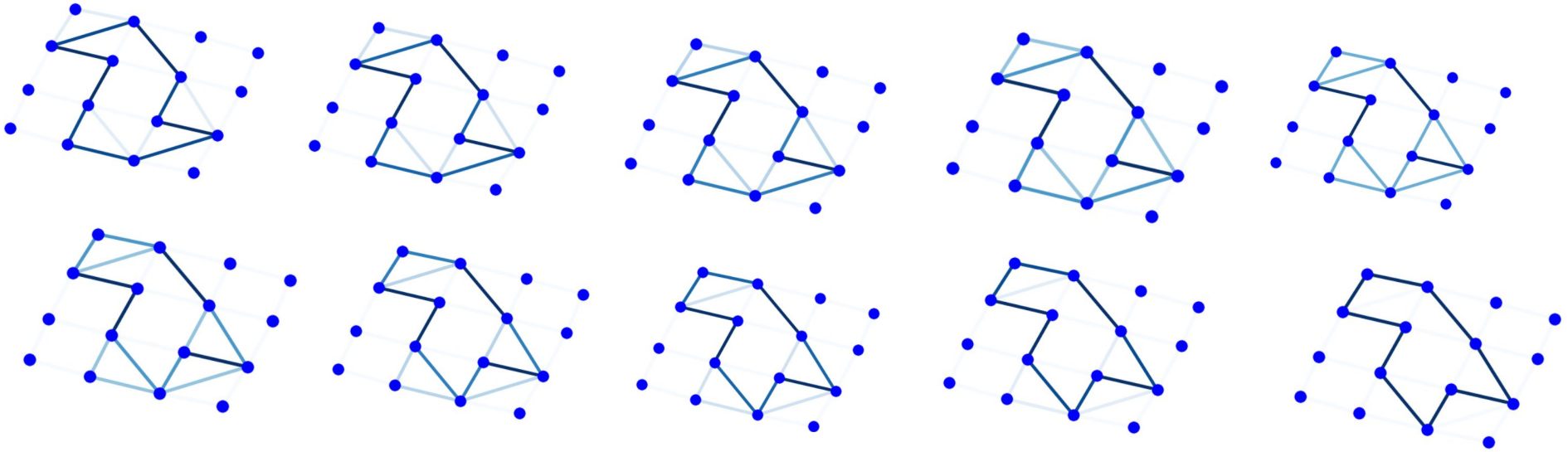






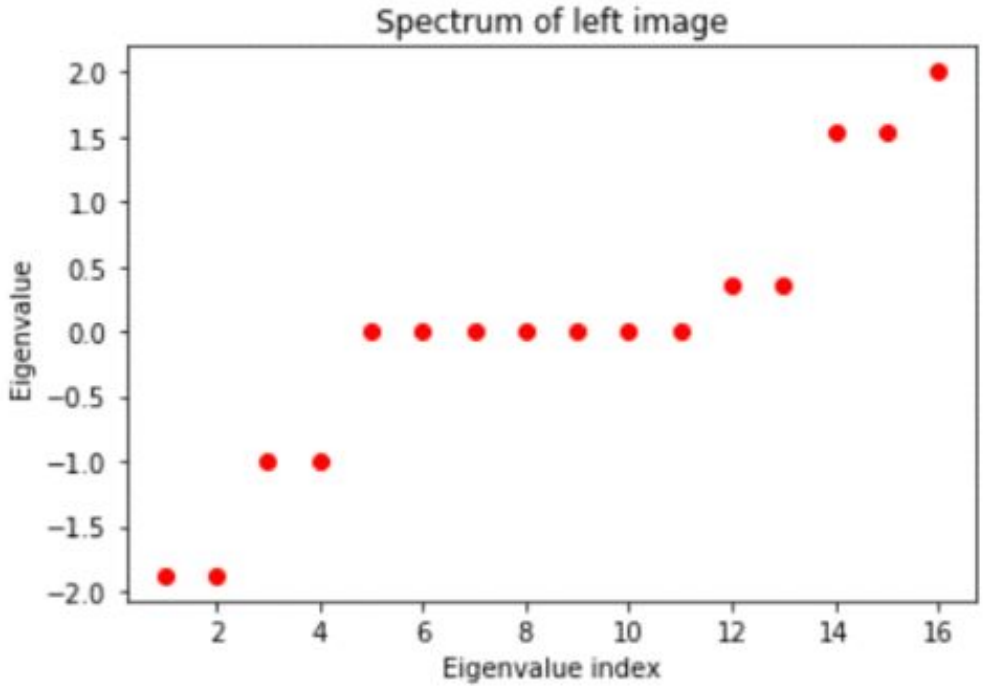


Transition



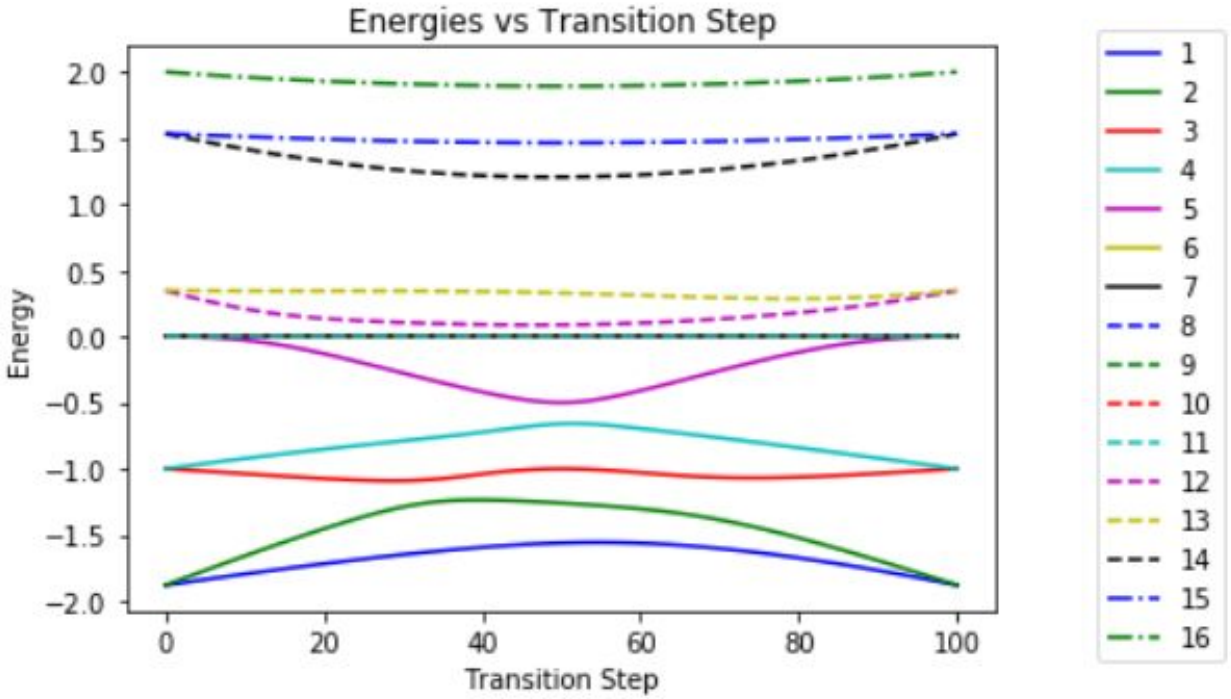
Spectra Transition

gif displays evolution of the spectrum



16 states due to 16x16 adjacency matrix

Energy levels in transition

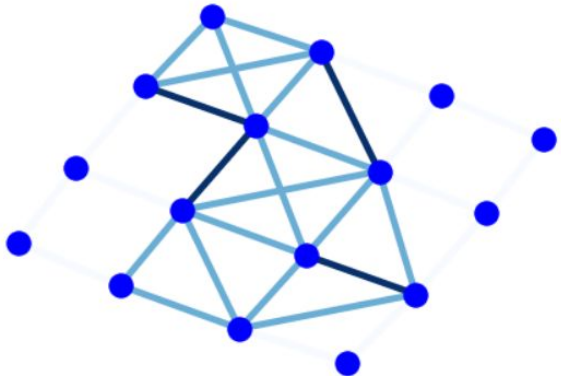


2nd Transition Path

Initia
l

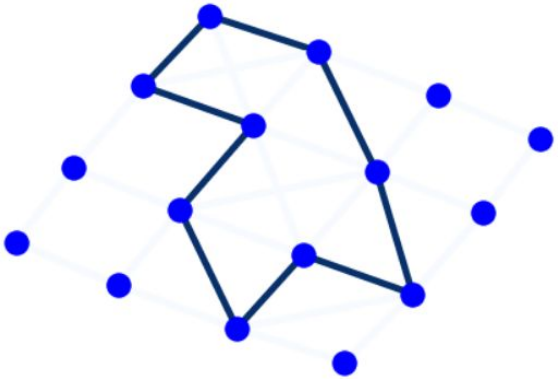
->

Middl
e



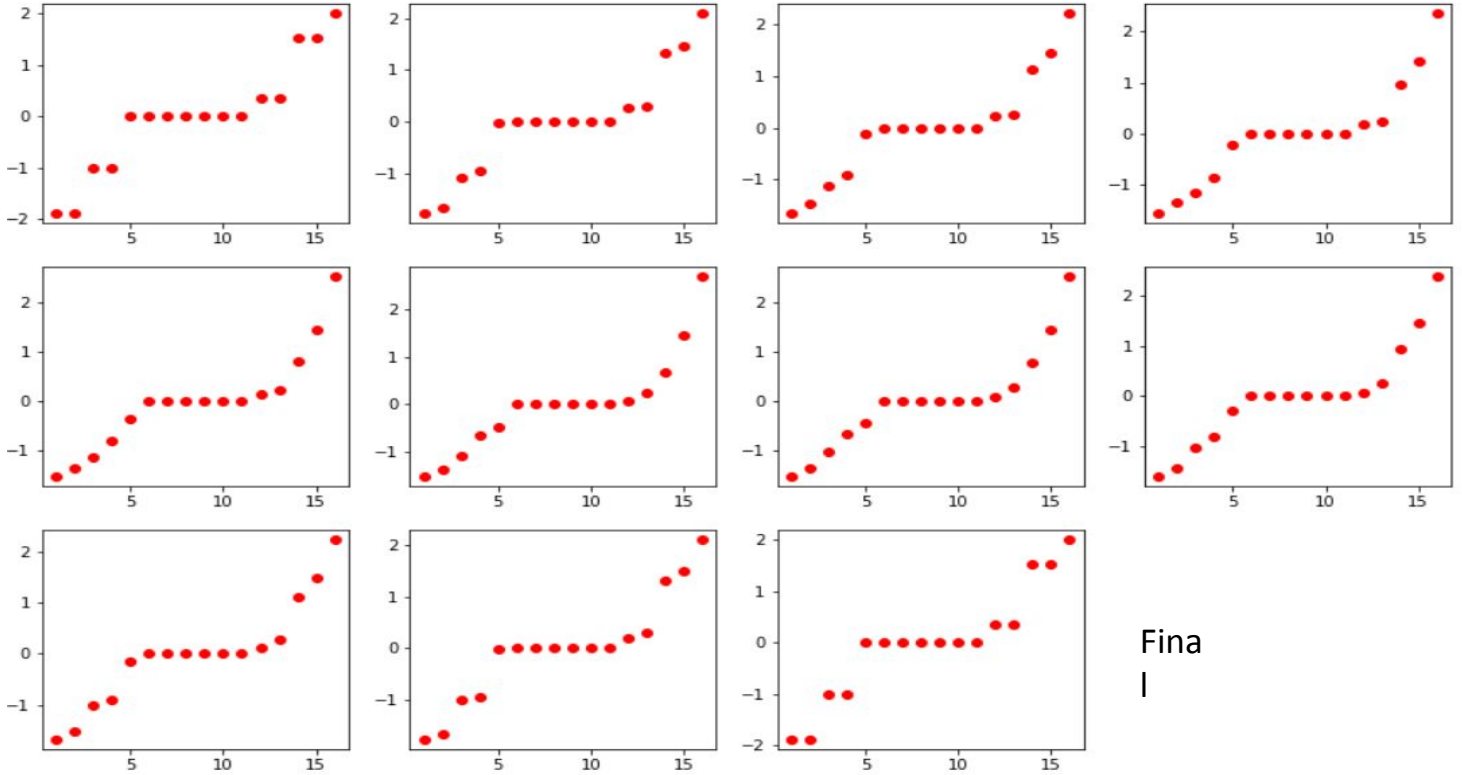
->

Fina
l



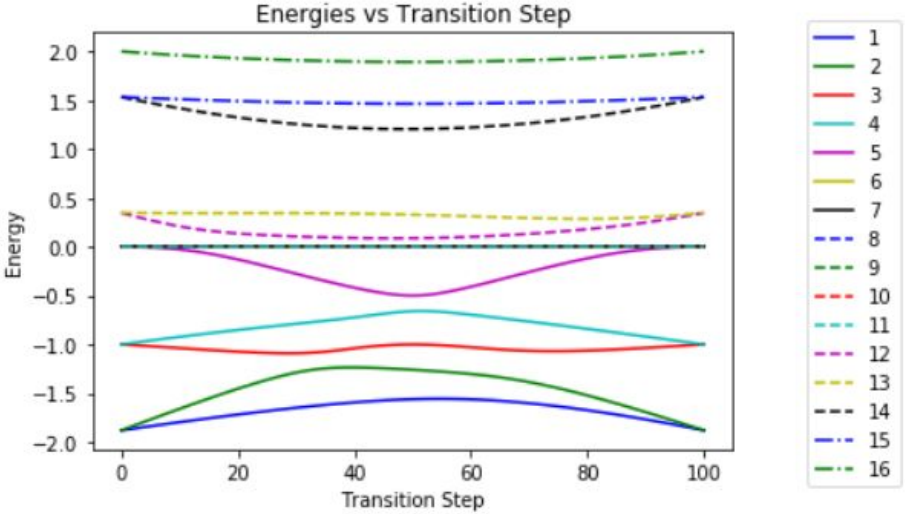
2nd Transition Spectra

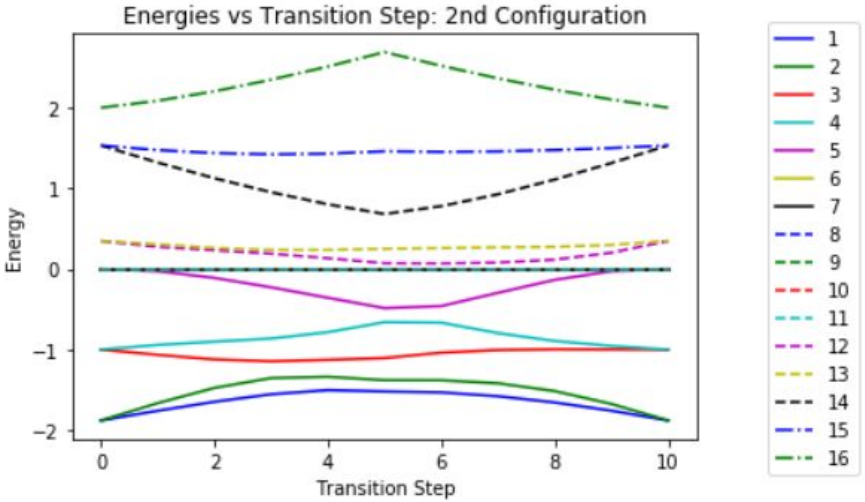
Initial



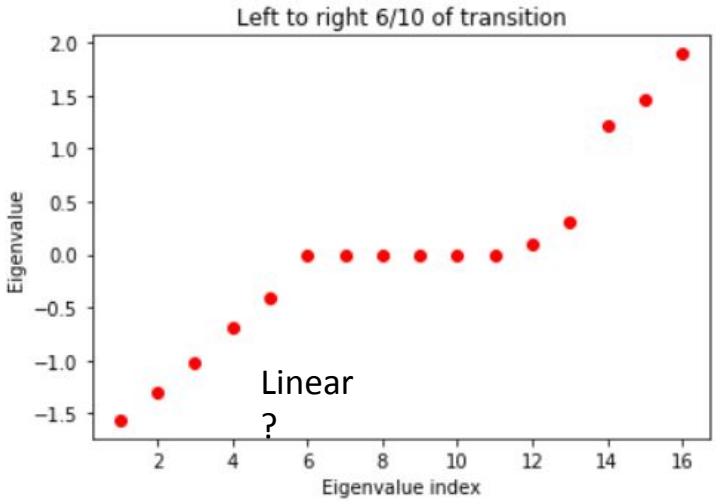
Fina
l

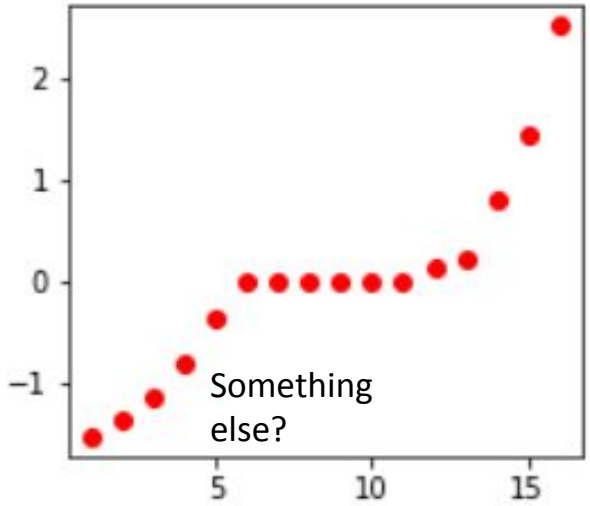
Energy Level Comparison





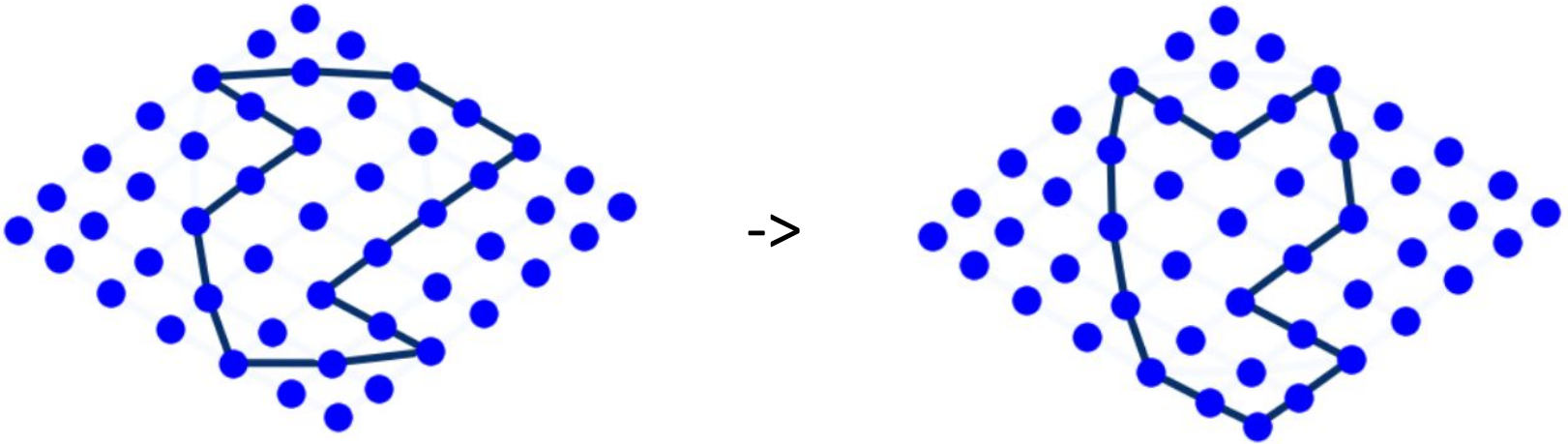
Intermediate Spectra Comparison



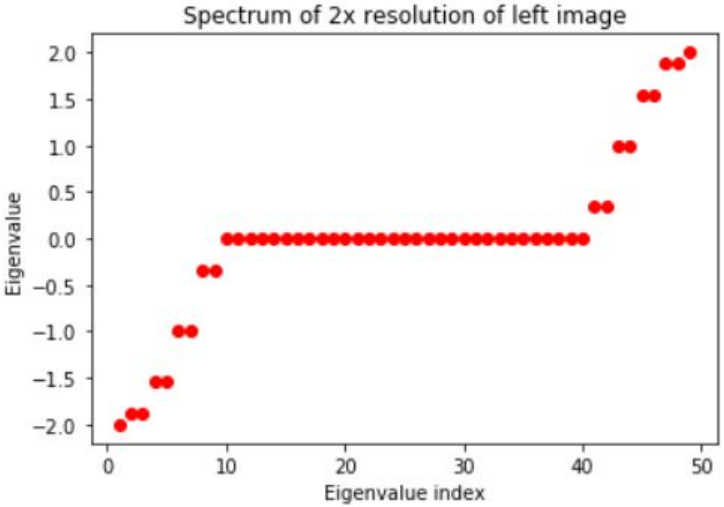


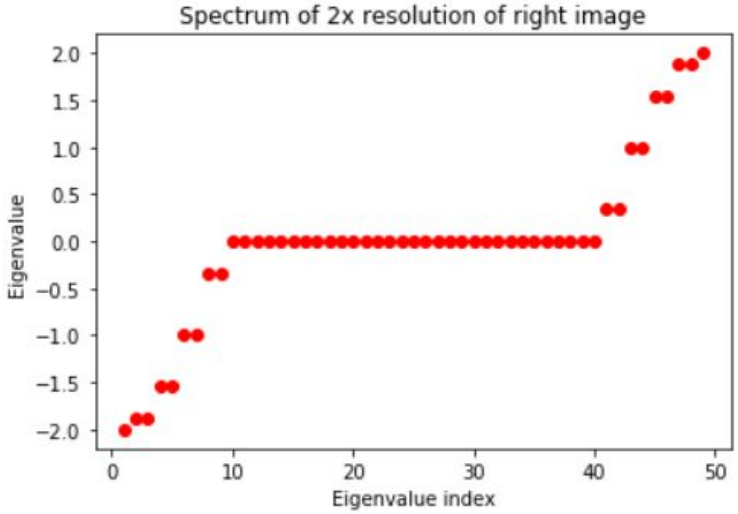
Investigate these transitions with more nodes = larger resonator setup

Larger Layout

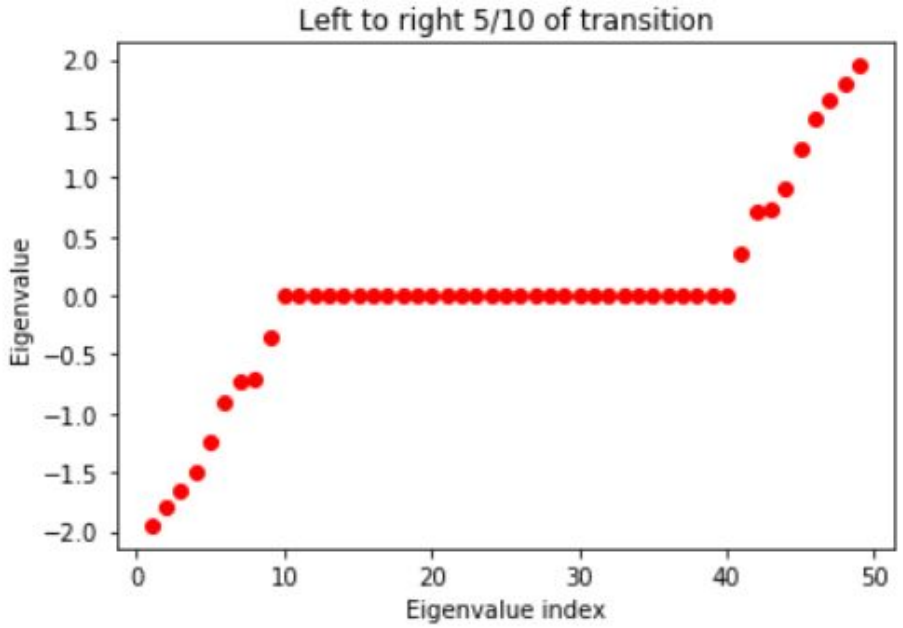


Isospectrality verified

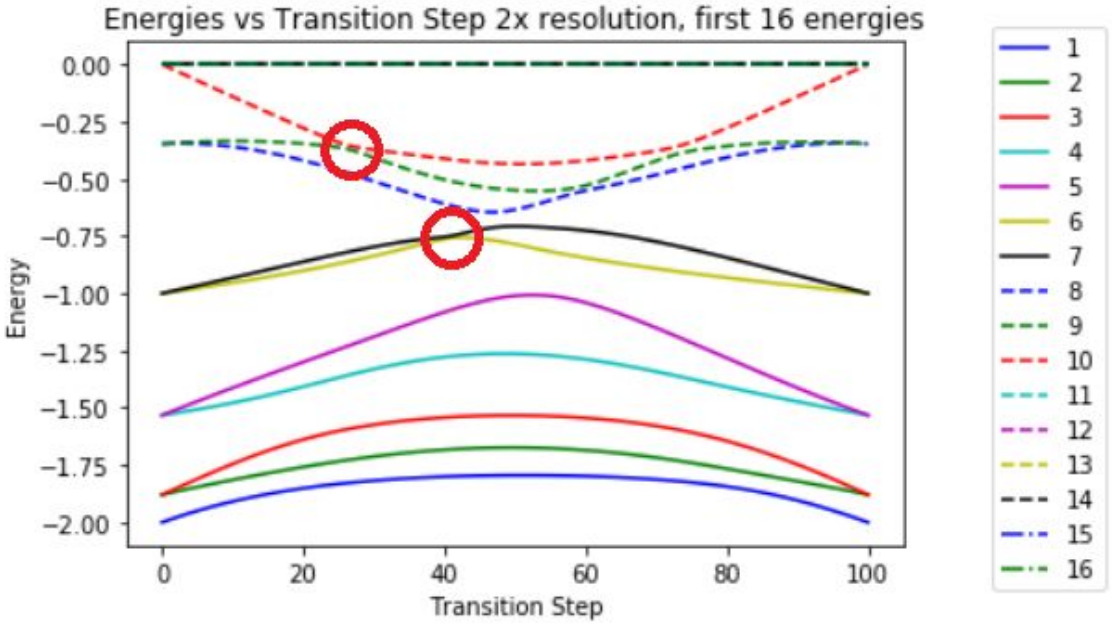




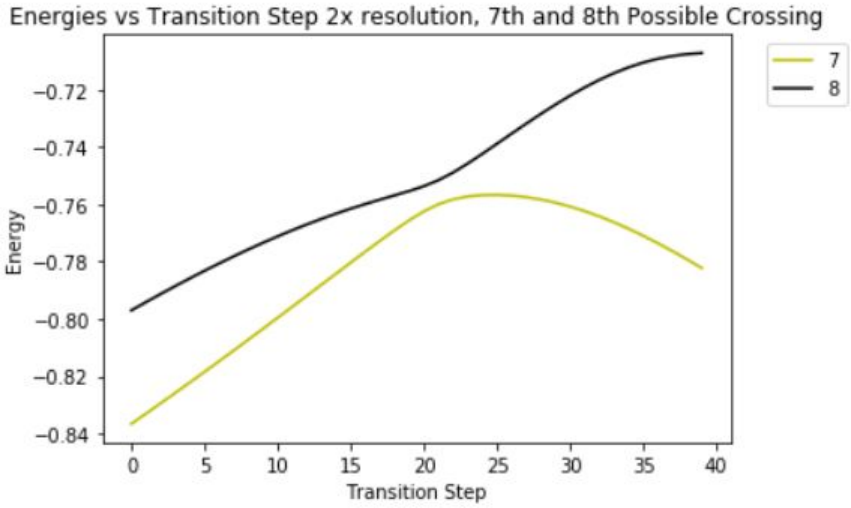
Suspected trend destroyed

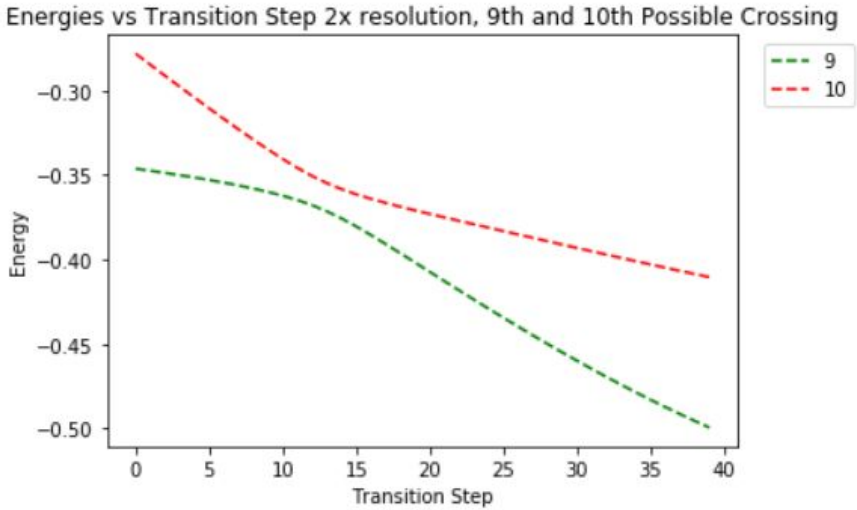


Possible energy level crossings?



Possible energy level crossings?





Future Work

- Explore coupling resonators to qubits
- Explore higher oscillator modes
- Explore nonlinear interactions
- Explore more techniques motivated by the literature

References

[1] Kollár, A.J., Fitzpatrick, M. & Houck, A.A. Hyperbolic lattices in circuit quantum electrodynamics. *Nature* **571,** 45–50 (2019).

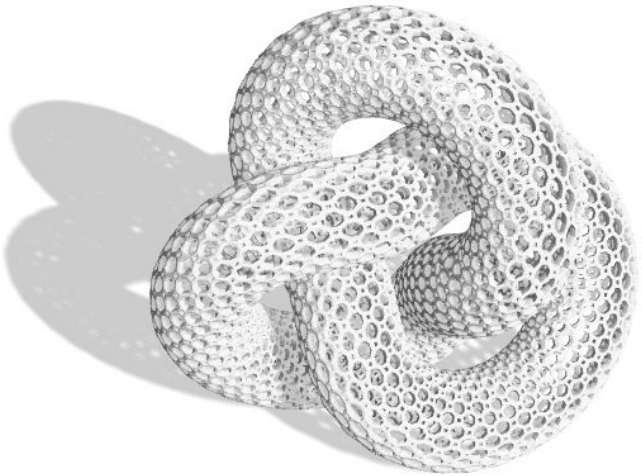
[2] Hafezi, M., Adhikari, P. & Taylor, J.M. Engineering three-body interaction and Pfaffian states in circuit QED systems. *Physical Review B* **90**, (2013).

[3] Hafezi, M., Mittal, S., Fan, J. *et al.* Imaging topological edge states in silicon photonics. *Nature Photon* **7,** 1001–1005 (2013).

[4] Hafezi, M., Demler, E., Lukin, M. *et al.* Robust optical delay lines with topological protection. *Nature Phys* **7,** 907–912 (2011).

[5] Moon, C., Mattos, L., Foster, B. *et al.* Quantum phase extraction in isospectral electronic nanostructures. *Science* **319**, 782-787 (2008).

NYU

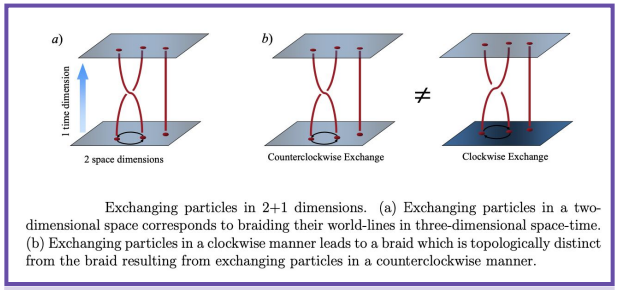


ARAHANT ASHOK KUMAR

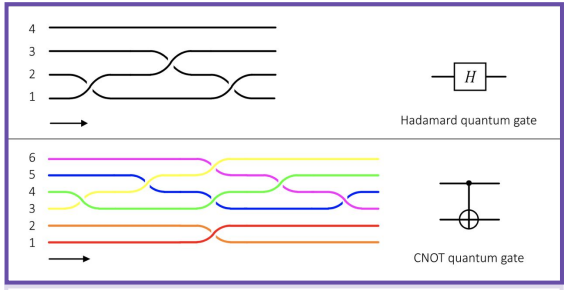
TOPOLOGICAL QUANTUM COMPILER

TOPOLOGICAL QUANTUM COMPUTING (TQC)

- In TQC, Quasiparticles such as Majorana fermions are braided in specific patterns and sequences to obtain Quantum gates
- Each braiding involves 2 particles being wrapped around each other…



Exchanging particles in 2+1 dimensions. (a) Exchanging particles in a two-dimensional space corresponds to braiding their world-lines in three-dimensional space-time. (b) Exchanging particles in a clockwise manner leads to a braid which is topologically distinct from the braid resulting from exchanging particles in a counterclockwise manner.



- In TQC, all the quantum gates depend only on the braid pattern. This would heavily suppress errors already at the level of the hardware, with little need for resource intensive quantum error correction.
- Physically, each of these braids are realised through movements of these particles on a Nanowire (Quantum Hardware)

CLASSICAL COMPILER

Code execution in current classical computers

- Compiler - Source code to Assembly language
  - Lexer - text into tokens
  - Parser - tokens into a Parse Tree
  - Parse tree into Intermediate Code
  - Intermediate code to Assembly language
- Assembler - assembly language to Object files
- Linker - links multiple object files and strings them together into ONE executable file
- Loader - Loads the executable onto the main memory for execution

(TOPOLOGICAL) QUANTUM COMPILER

<table>
<tr><td colspan="3">TQC QUANTUM COMPILER ARCHITECTURE</td></tr>
<tr><td rowspan="3">Quantum Compiler</td><td colspan="2">TQC Library</td></tr>
<tr><td>Restricted APIs</td><td>Modules</td></tr>
<tr><td colspan="2">System calls</td></tr>
<tr><td rowspan="3">Programmable Quantum Material</td><td colspan="2">Assembly instructions</td></tr>
<tr><td colspan="2">Programmable QM</td></tr>
<tr><td>H/W config</td><td>APIs</td></tr>
<tr><td colspan="2">Quantum Material (QM)</td></tr>
</table>

TOPOLOGICAL QUANTUM COMPILER STAGES

- Preprocessing - It converts the given inputs into machine readable formats
  - Nanowire - adjacency matrix
  - Braid sequence - slices of pair-wise braid sequences
- Topological Quantum Compiler Algorithm
  - Nanowire Rules - Physical constraints, Nanowire state, etc.
  - Phases - Validation and Braiding
- Output - Particle movement sequences, Nanowire state matrix

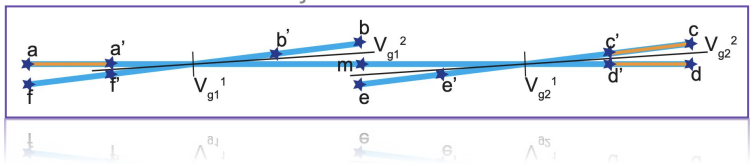
PRE-PROCESSING - NANOWIRE

Given a Nanowire structure, an Adjacency matrix is constructed, which is used to determine the paths of the movements.

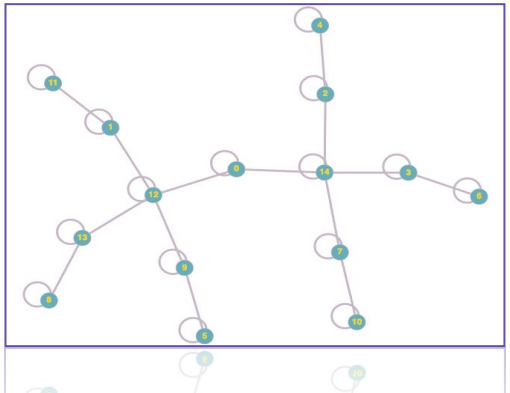
Nanowire Adjacency Matrix

1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	1	0	1
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	1	0	0	0	1	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	1	1	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1	1	0	0
0	1	1	0	0	0	0	0	0	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

Double-X junction Nanowire



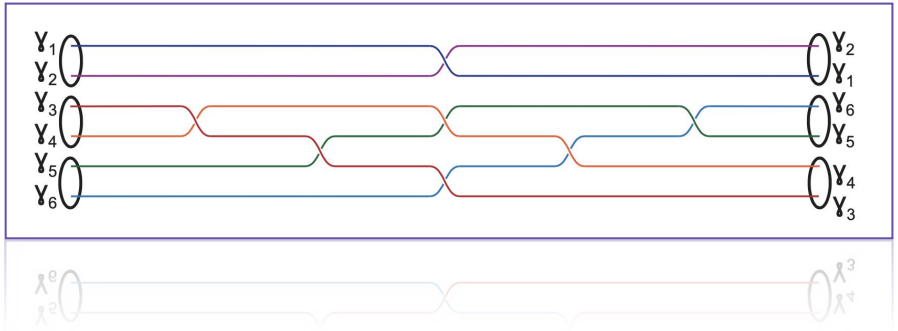
Reconstructed Nanowire Graph



PRE-PROCESSING - BRAID SEQUENCE

- Given a Braid pattern for a Quantum gate, it's processed into a sequence of braids of 2 particles.
- Each slice of this sequence is an atomic operation.

Braiding pattern for a 2-Qubit CNOT gate



Braid sequence

(3, 4)
(3, 5)
(1, 2)
(4, 5)
(3, 6)
(4, 6)
(5, 6)

Initial Particle positions

The initial positions of the particles on the Nanowire, in the above example, are [a, a', c, c', d, d']

ALGORITHM - RULES

- Nanowire State validity - Each braiding must result in a valid nanowire state
- Particle-Zero mode isolation - Two particles from different zero modes cannot occupy adjacent positions on the Nanowire during the braiding operation.
- Non-intersecting zero-modes - The non-participating paired zero-mode particles cannot be on the same crossing.
  - This would involve voltage regulation.
- Braiding Rotation - Clockwise and anti-clockwise braiding.
- Braiding Sequence - The entire braiding procedure follows the given sequence
- Atomic Braiding operation - Each braiding slice involves 2 particles

- Braiding Concurrency limits - limit on the # concurrent braiding operations
- Intermediate positions - Every braiding operation puts the particles back in their respective final positions.

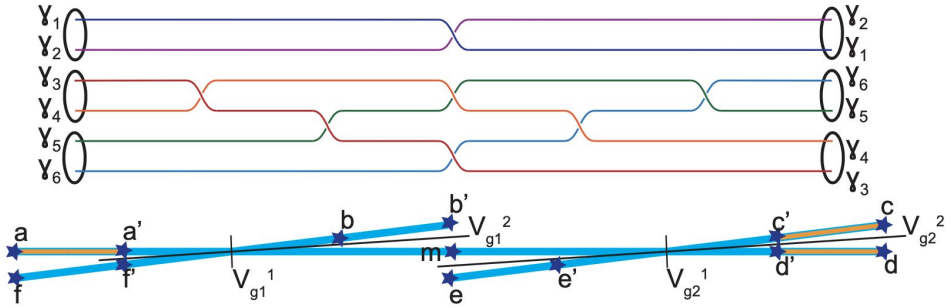
ALGORITHM - VALIDATION PHASE

- Validate Final Positions - Retrieve and Validate the expected final positions for the particles, after the Braiding operation is completed.
- Validate Intermediate Positions - Retrieve, validate and Rank the potential Intermediate positions for the participating particles.
  - 1st particle that moves gets the 1st empty branch. Later, specify rules/restrictions.
- Ranking multiple positions - If there are more than one free branch, there can be multiple final positions, which are ranked based on their…
- Validate Resulting State - Nanowire validation algorithm which returns a score for every expected final/ intermediate position.

ALGORITHM - BRAIDING PHASE

- Retrieve involved branches - Retrieve the list of branches and its positions involved in the movement of the particle.
- Get Shortest Path - the shortest path for a particle from its current position to the given (valid) final/ intermediate position using Dijkstra's algorithm.
- *Get Voltage Changes - If braiding involves particles from different zero modes, perform necessary voltage gate changes.

OUTPUT - NANOWIRE STATE MATRIX



γ_1	a	a	a	a	a	a	a	a	**b**	b	**a'**	a'	a'	a'	a'	a'	a'	a'	a'	a'	a'	a'	a'	a'
γ_2	a'	a'	a'	a'	a'	a'	a'	**f**	f	**a**	a	a	a	a	a	a	a	a	a	a	a	a	a	a
γ_3	c	c	**m**	m	m	m	m	m	m	**d**	d	**e**	e	e	e	e	e	e	e	e	e	e	e	e
γ_4	c'	**e**	e	**c**	c	c	c	c	c	c	c	c	c	c	**d**	d	d	d	d	**e'**	e'	e'	e'	e'
γ_5	d	d	d	d	d	d	d	**c'**	c'	c'	c'	c'	c'	**m**	m	**c**	c	c	c	c	c	c	**m**	**d'**
γ_6	d'	d'	d'	d'	d'	**e**	e	e	e	e	**m**	m	**e'**	e'	e'	e'	e'	**m**	m	m	m	**d**	d	d
V_{g2}^1	o	o	o	o	o	o	**s**	s	**o**	o	o	o	o	o	o	o	**s**	s	**o**	o	o	o	o	o
V_{g2}^2	o	o	o	o	**s**	s	**o**	o	o	o	o	o	o	o	o	o	o	o	**s**	s	**o**	o	o	o

OUTPUT - NANOWIRE STATE MATRIX

Nanowire state matrix for Braiding between particles (1, 2)

1	a	a	a	a'	f'	f'	f'	f'	f'	a'
2	a'	b'	b'	b'	b'	b'	a'	a	a	a
3	d	d	d	d	d	d	d	d	d	d
4	c	c	c	c	c	c	c	c	c	c
5	c'	c'	c'	c'	c'	c'	c'	c'	c'	c'
6	d'	d'	d'	d'	d'	d'	d'	d'	d'	d'

x - empty positions

o - particles

NYU

FUTURE WORK

- Integrate RULES
  - Voltage regulation
  - Direction
- Optimisation - Intermediate states
- Optimisation - Concurrency
- Optimisation - Hardware Cost (sequence interchangeability)
- Fusion
- Preprocessing - Braiding pattern to Braiding sequence

FUTURE APPLICATIONS

- Quantum Computing
  - Potentially error-free Quantum gates and circuits
  - Realisation of Topological Quantum Compiler for Finance
- Cryptography
  - Braiding-based cryptography
- Knot theory
  - Jones polynomial
  - DNA recombination

Questions?

Thank you

GitHub

aak700@nyu.edu

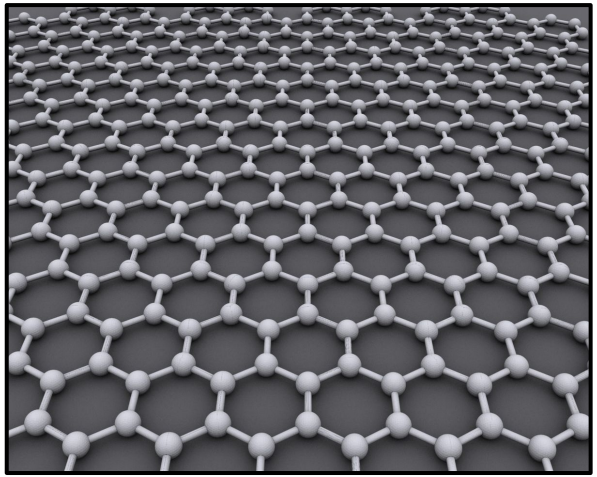
Guardians Final Presentation

Eric Berg, Nikki Ebadollahi, Abigail Martucci

Topic Overview

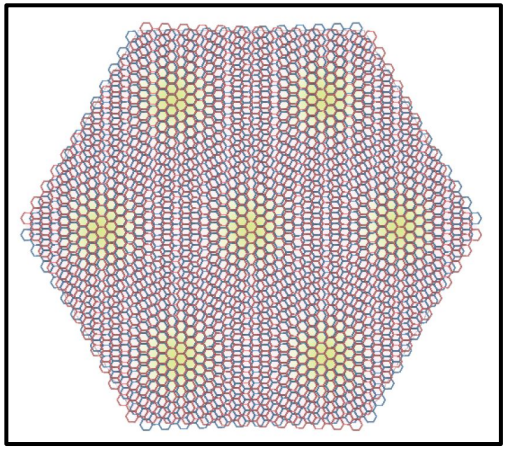
- Graphene - What is it and why it’s important?
- Graphene Applications: Twisted bilayer graphene.
- Graphene Exfoliation - From a crystalline graphite to a monolayer graphene.
- Automation of graphene exfoliation and 3D Design of the model
- Electrical assembly and coding to automate exfoliation process: Abigail and Nikki
- Microscope development and imaging automation: Eric
- Conclusion: How all 3 components work together

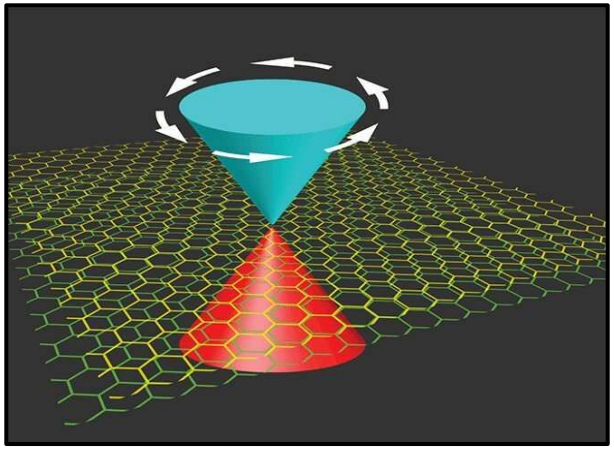
Graphene



- Sheet of carbon atoms just one atom thick.
- Two dimensional system made of carbon atoms in a honeycomb lattice.
- Reducing the thickness of graphene opens new possibilities.
- Graphene has extremely high electrical current density
- Graphene has high thermal conductivity

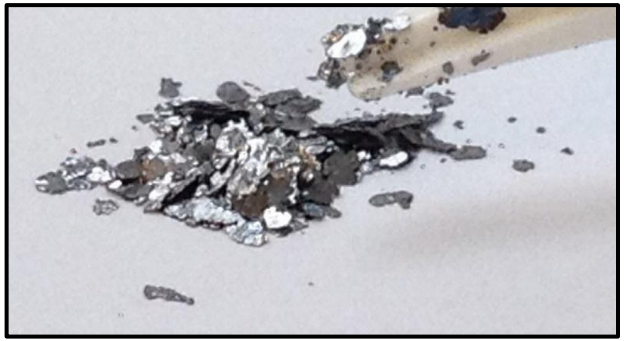
Graphene is used for Superconductivity!

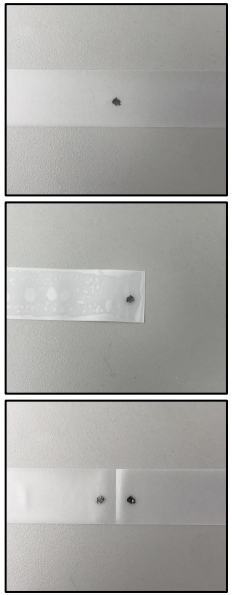


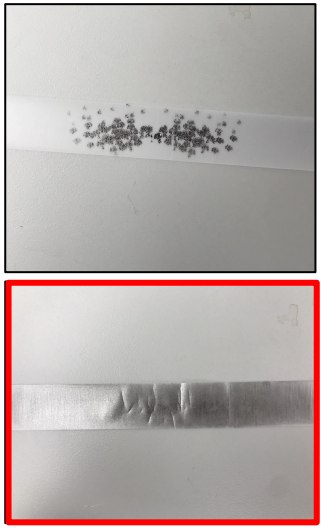


A small relative twist of 1.1 degree between the layers of bilayer graphene can induce superconductivity.

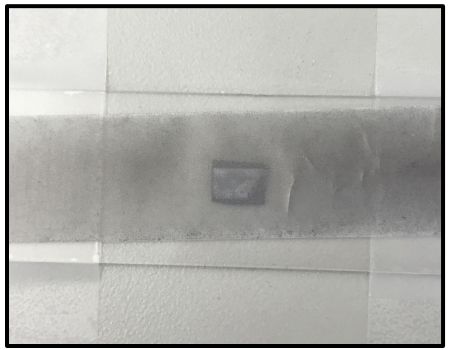
Step 1_Tape Exfoliation

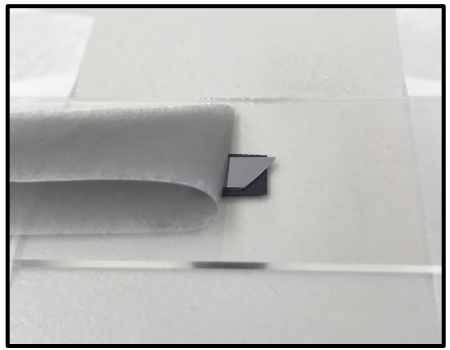


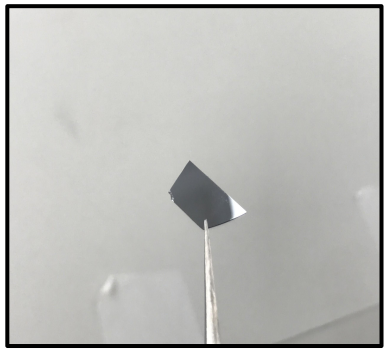


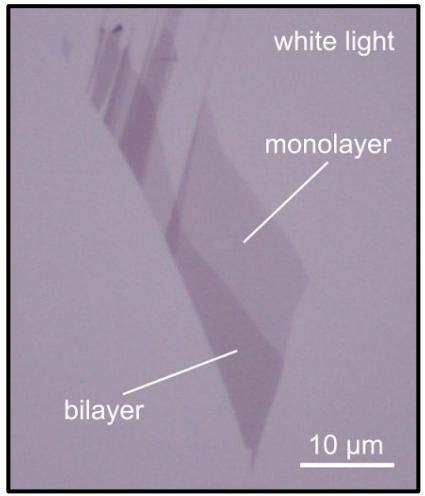


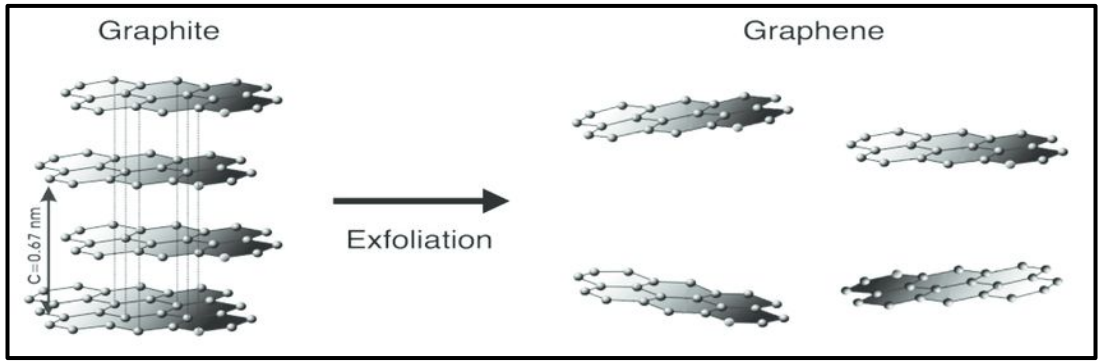
Step 2_Graphene Exfoliation



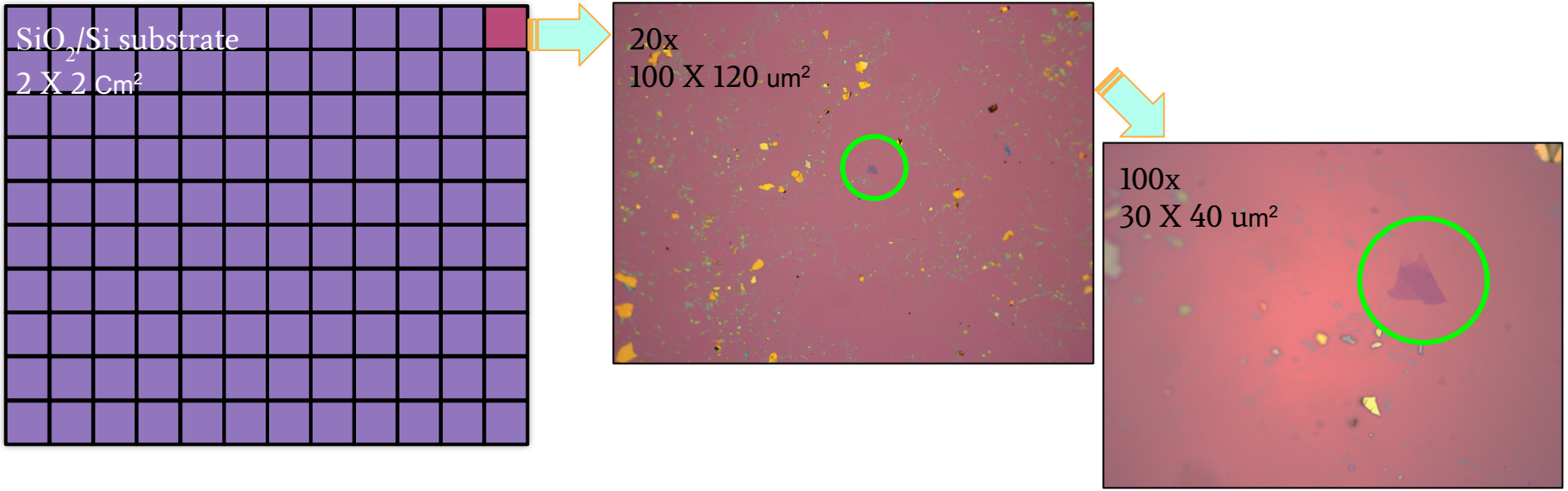




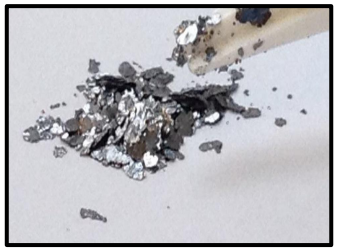


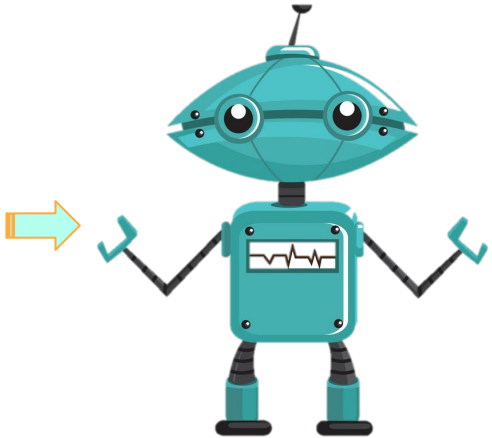


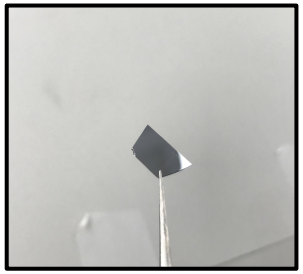
Step 3_Imaging

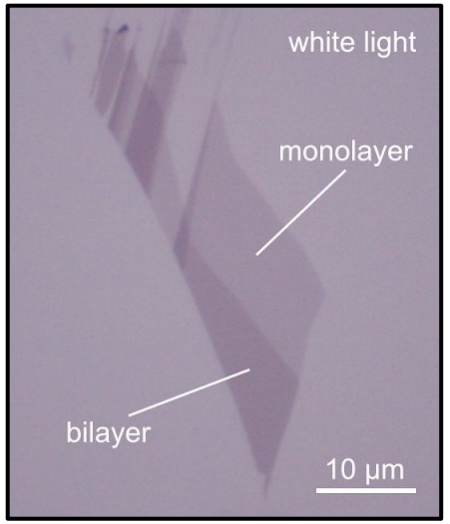


Automation of Exfoliation of Graphene



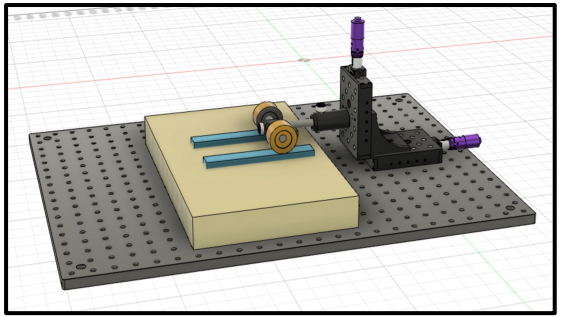


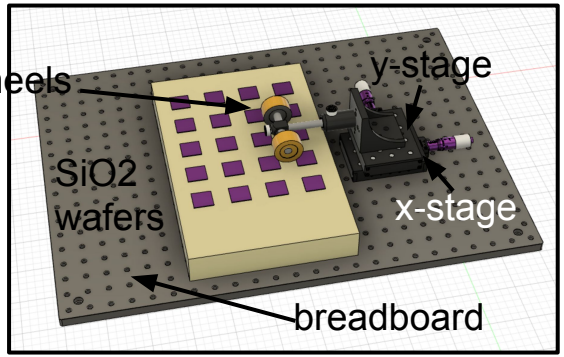


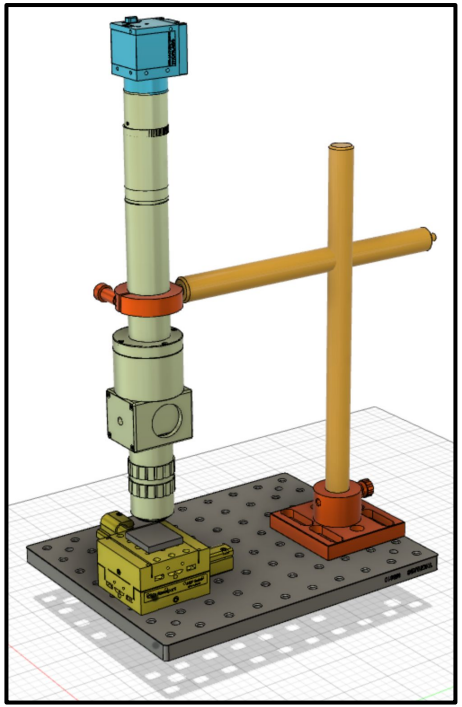


1) Automation of tape peeling process: Nikki
2) Automation of exfoliation process: Abigail
3) Automation of imaging process: Eric

Design Structure of Exfoliation Automation



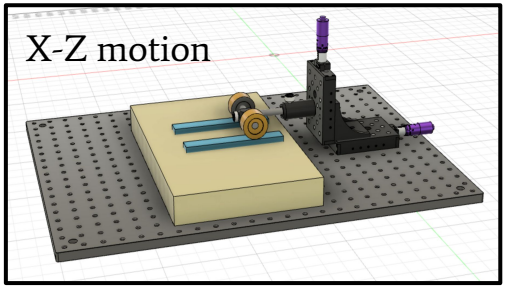


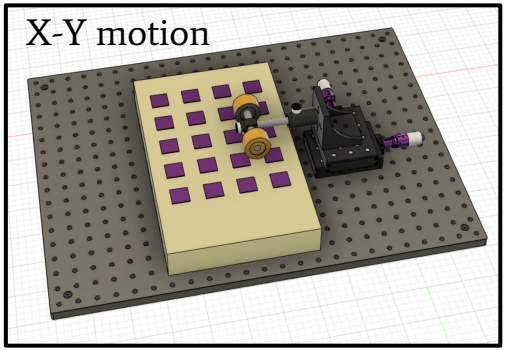


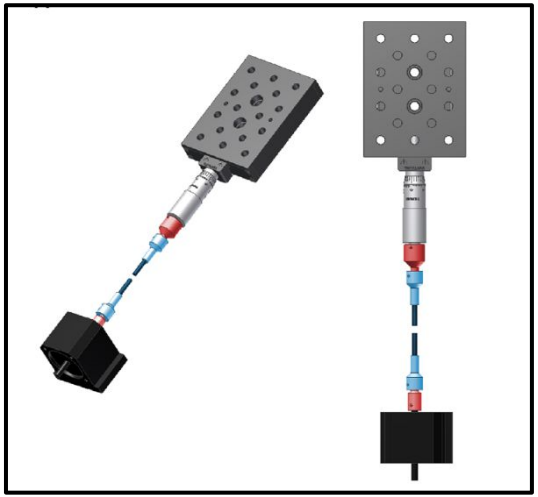
Graphene Exfoliation Automation

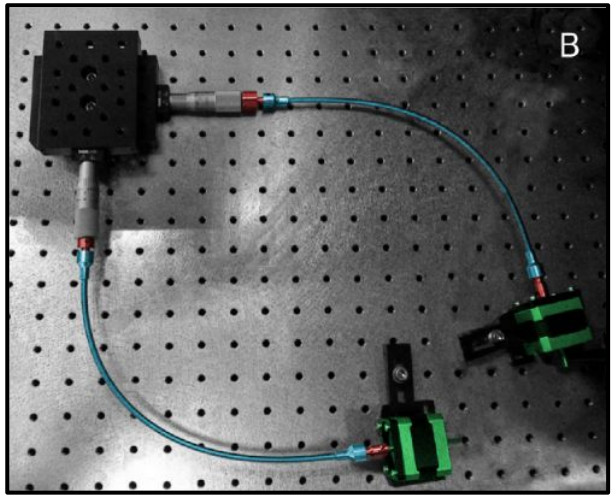
Abigail Martucci and Nikki Ebadollahi

Mechanical to Electrical





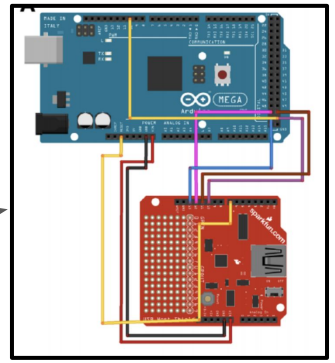


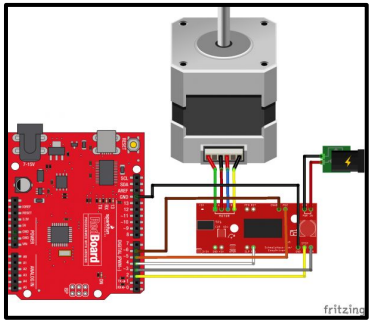


Connect ThorLabs PT1 linear translator with micrometer to stepper motor via flexible shaft. Cylindrical couplers are in red and flexible shafts are blue.

Electrical Overview

1) Install firmware onto Arduino
2) Wire the system
   a) USB Host Shield to Arduino Mega Connection
   b) USB Host Shield & Motor to Driver Connection
3) Communicate with stages
4) Testing of motion
   a) Speed & Step Sizes
   b) Back & Forth Motion





2a. Wiring of Shield to Arduino

a. Connect pin 7 to reset on the shield.

b. Connect reset on the shield to reset on the Mega.

c. Connect GND and VIN on the shield to GND and VIN on the Mega.

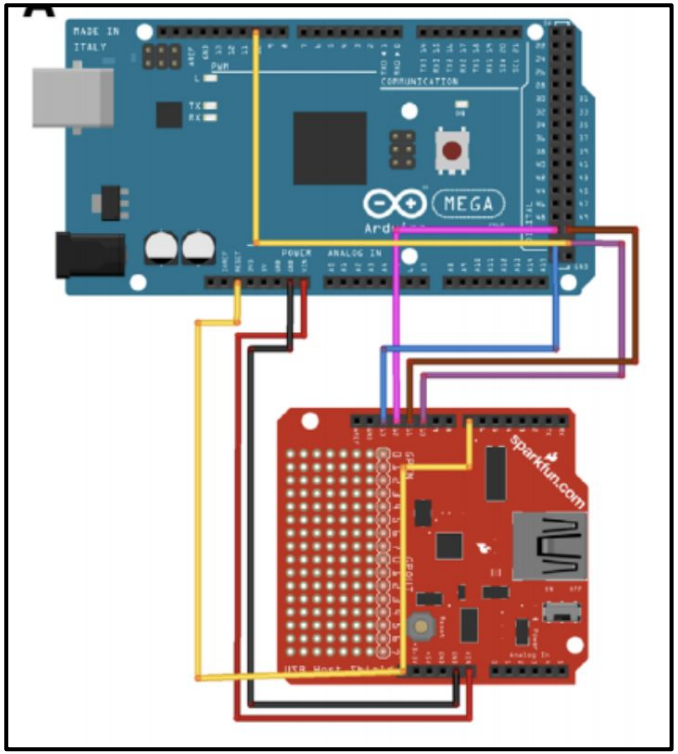
d. Connect pin 13 on the shield to pin 52 on the Mega.

e. Connect pin 12 on the shield to pin 50 on the Mega.

f. Connect pin 11 on the shield to pin 51 on the Mega.

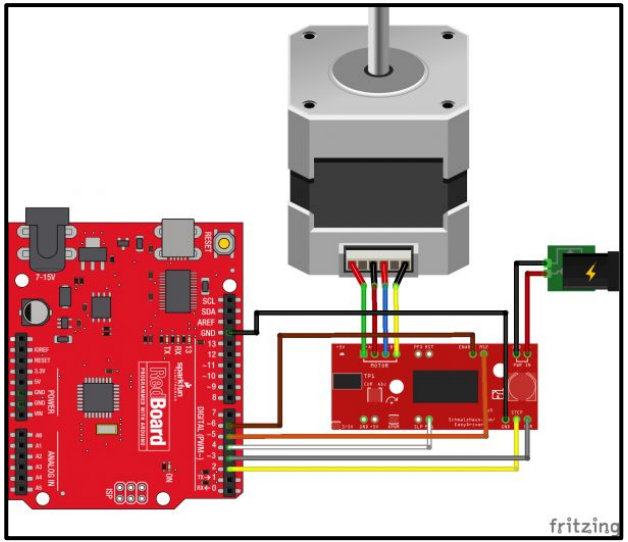
g. Connect pin 10 on the shield to pin 53 on the Mega.

h. Connect pin 10 on the Mega to pin 53 on the Mega

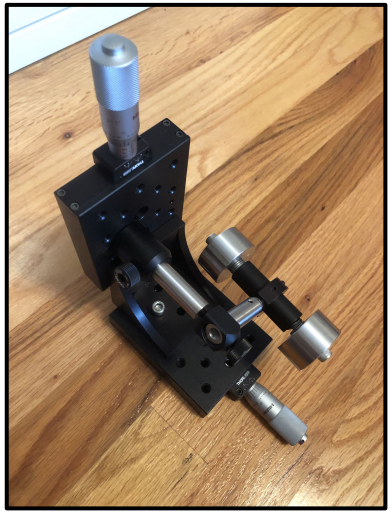


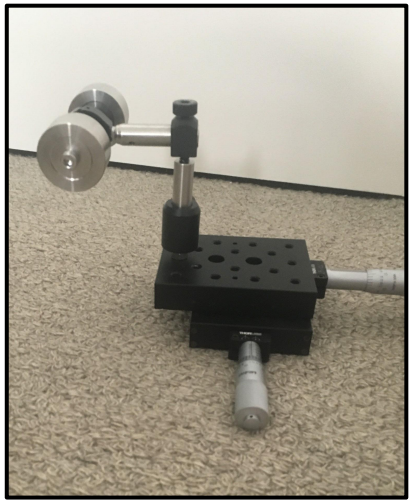
2b. Wiring of Shield and Motor to Driver

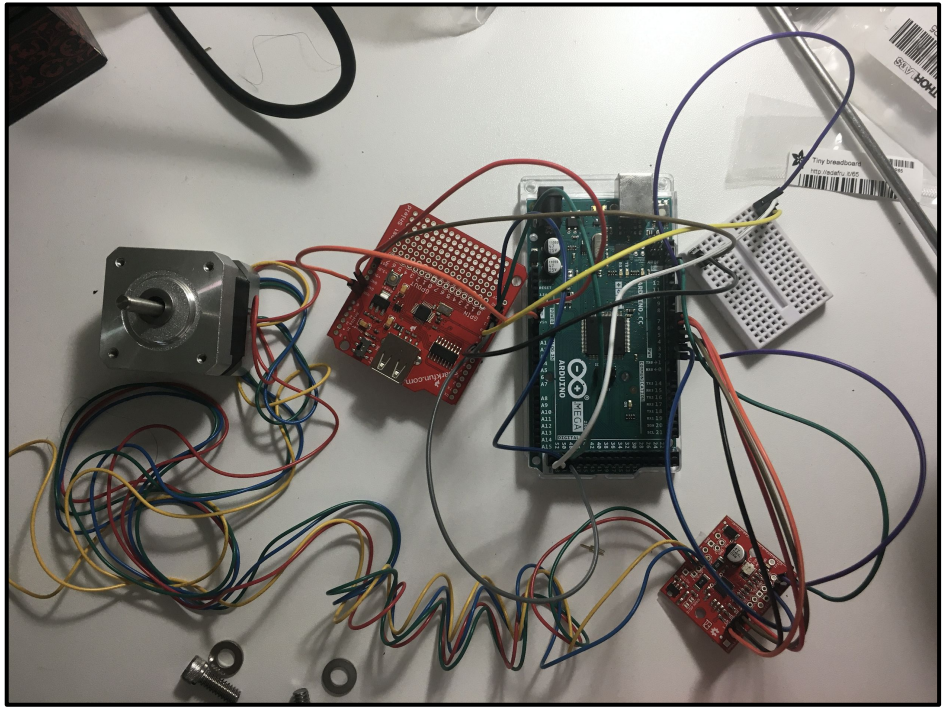
a. Connect pin 2 on the shield to STEP on driver
b. Connect pin 3 on the shield to DTR on driver
c. Connect pin 4 on the shield to MS1 on driver
d. Connect pin 5 on the shield to MS2 on driver
e. Connect pin 6 on the shield to ENABLE on driver
f. Connect GND on the shield to GND on driver
g. Connect green, red, blue and yellow wires from motor to the motor pins on the driver from left to right in the order given



Our Assembly







3. Communicate with Stage

- Open the connectOpenStage.m file and replace the existing port name with your port. Save and close. Issue the OS_beep command.
- Attach a piece of tape to the spindle (like a little flag) to act as a clear visual indicator that the spindle has rotated.
- As a test, program spindle to rotate 360° & make sure it returns to start position:

  Gear Ratio Value = # of microns/rev

  ```
  while 1, OS_goto(635,1); pause(1), end
  ```

- Note: If you do not see 360 degree rotations:

A. Check your gear ratio value

B. Check that micro-stepping wires are hooked up

C. Make sure that you're not driving too much or too little current to the motor

4a. Test Motion- Speed and Step Size

- Check that the motor performs well at faster speeds. Set step size to full steps (OS_stepSize(1)).
- Check motions again -- Try faster motion speeds with OS_moveSpeed:
- Confirm this worked with OS_getInfo. This also is used to determined the max RPM asking from the motor (make sure this matches with value on spec sheet).

- Notes: If stalling or skipping well below that value (half or quarter of that value)
  A. Check your motor current (too high or low is bad)
  B. Consider more gentle acceleration regime by going half steps (this often leads to smoother motions & less skipping)

4b. Test Motion- Back and Forth Motion

- Choose some reasonable settings.
- Reset the Arduino and start issuing absolute motion commands:

  OS_goto(6350,'a') causes the motor to advance 10 full rotations in one direction

  OS_goto(0,'a') sets it back to the starting point.

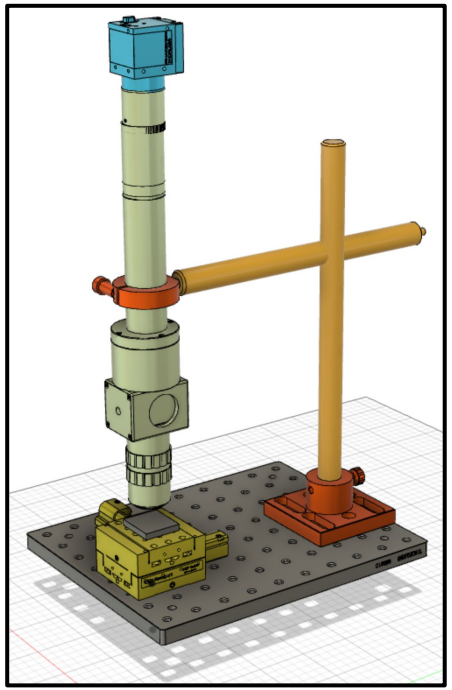
  `OS_zero;` makes the current position the zero location

- Try these out with different motions, directions and step sizes

Optical Microscope

Eric Berg

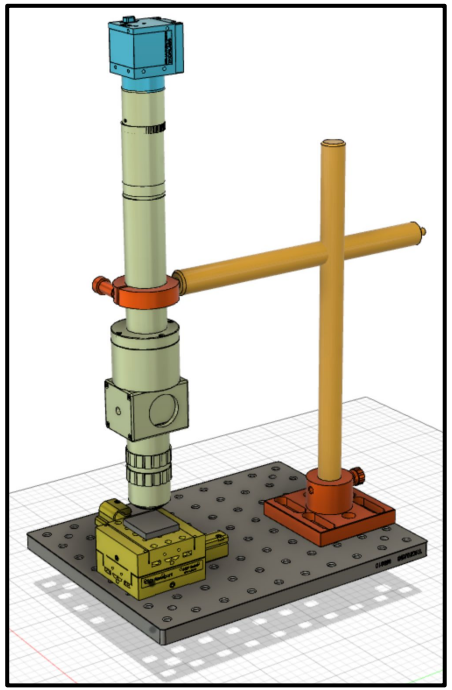
Imaging Automation

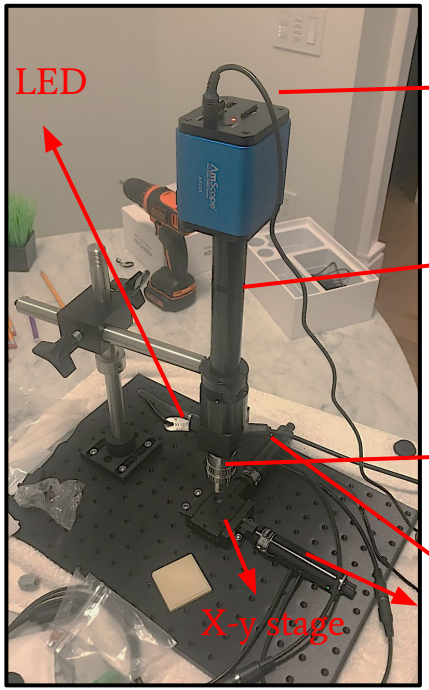


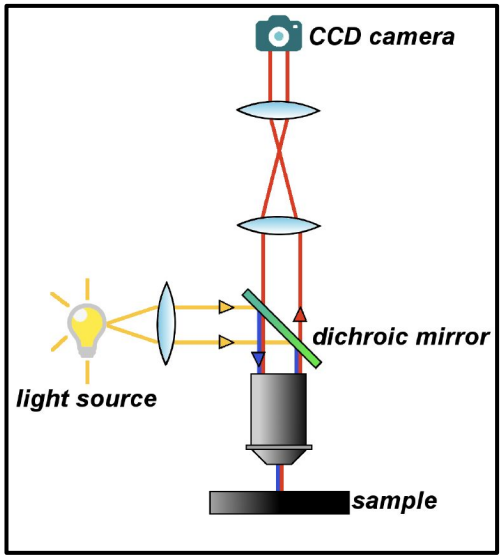
Goal is to create an imaging setup that can automatically scan a sample and produce a high-resolution image.

1. Optical microscope
2. Micrometer 2D-motion of x-y stage
3. High resolution imaging from a large scale wafer

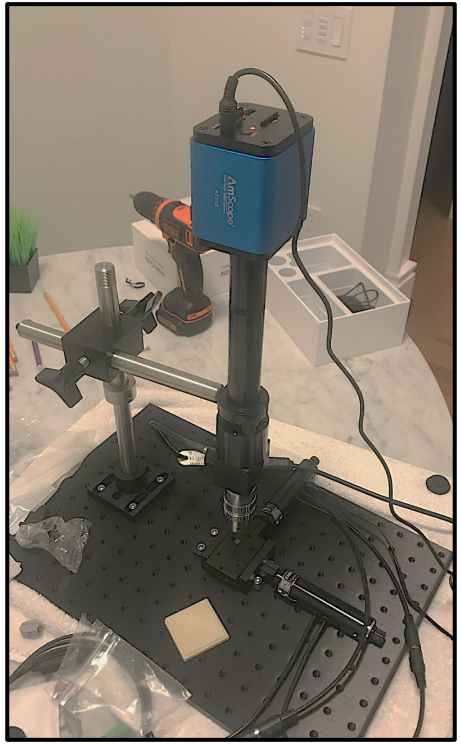
1. Optical Microscope

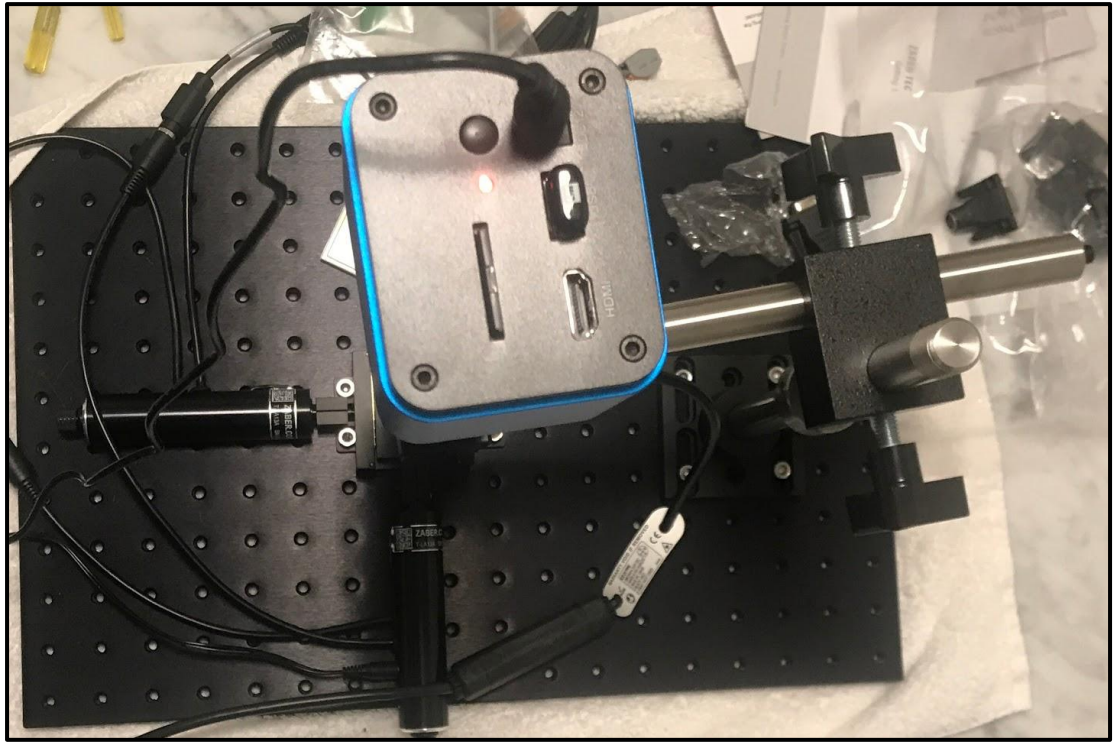






1. Optical Microscope





2. Micrometer 2D-motion of x-y stage

In [34]:
```
from zaber_motion import Library
from zaber_motion.binary import Connection
from zaber_motion import Units
import math

Library.enable_device_db_store()

# The rest of your program goes here
```

In [42]:
```
flakeWidth = 10 #mm
flakeHeight = 3 #mm
stepDownSize = 2 #mm per every horizontal sweep
#NOTE - Always put the flake with objective ligned up at top left corner
#NOTE - Make the flakeWidth and flakeHeight a bit bigger than it actually is to account for tilt of flake

with Connection.open_serial_port("/dev/tty.usbserial-AK06IX95") as connection:
    connection.renumber_devices()
    device_list = connection.detect_devices()
    print("Found {} devices".format(len(device_list)))

    xAxis = device_list[0]
    yAxis = device_list[1]
    xAxis.home()
    yAxis.home()
    for i in range(math.ceil(flakeHeight/stepDownSize)):
        # Move to the flakeWidth mm position
        xAxis.move_absolute(flakeWidth, Units.LENGTH_MILLIMETRES)
        # Move back by flakeWidth mm
        #xAxis.move_relative(-flakeWidth, Units.LENGTH_MILLIMETRES)
        yAxis.move_relative(stepDownSize, Units.LENGTH_MILLIMETRES)

        xAxis.home()
        yAxis.move_relative(stepDownSize, Units.LENGTH_MILLIMETRES)
    xAxis.home()
    yAxis.home()
```

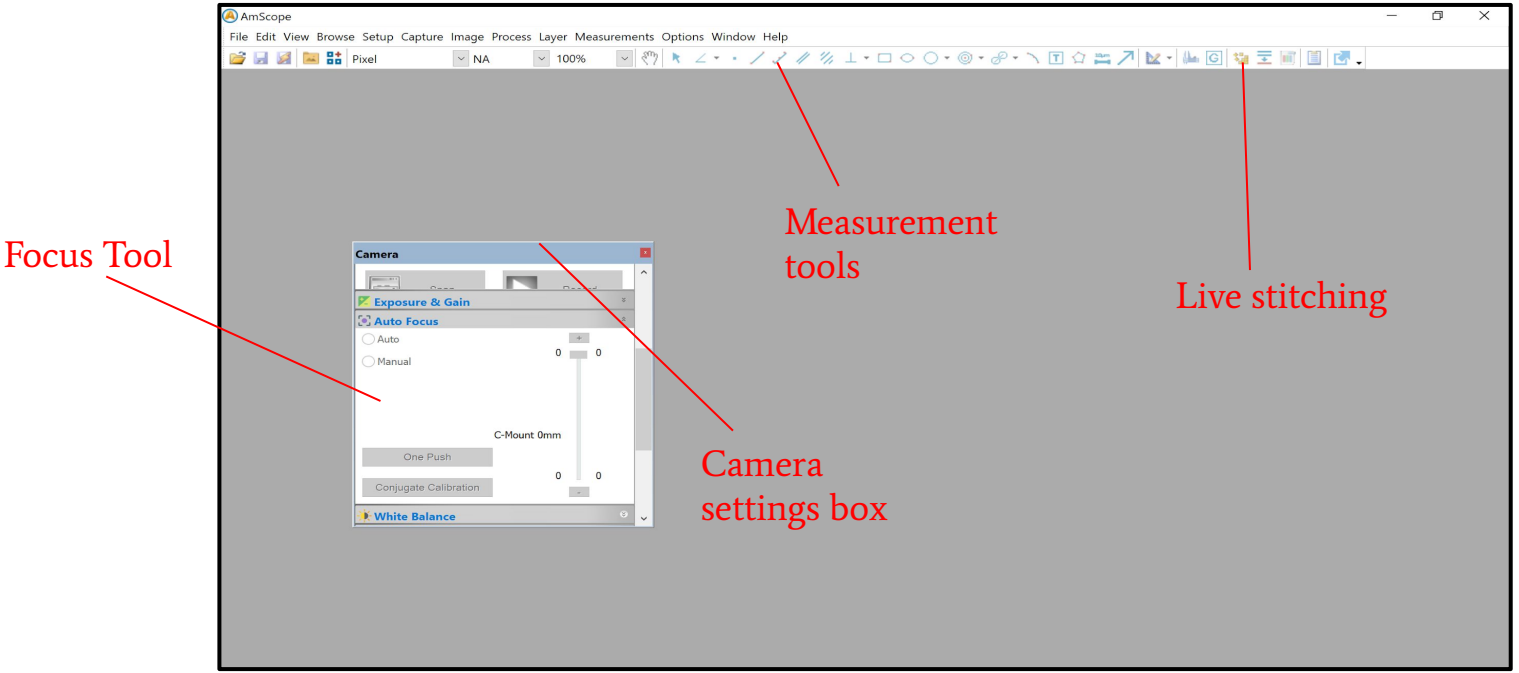
```
Found 2 devices
```

Actuators that run through python coding allow the x-y motion of the stage.

2. Micrometer 2D-motion of x-y stage

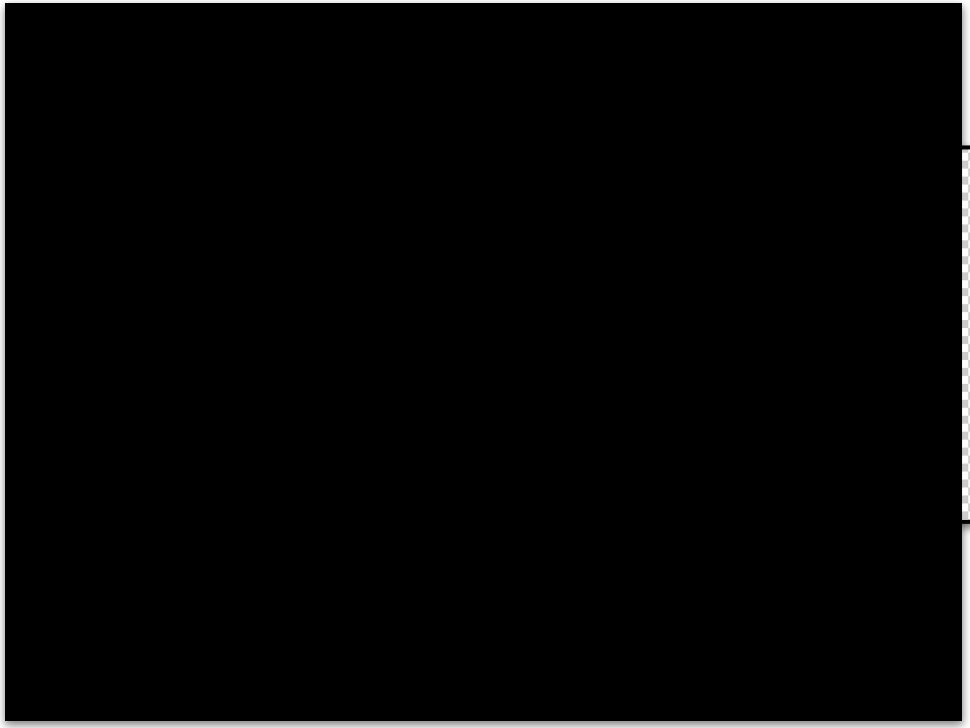
Stage motion after implementing python code on actuators.

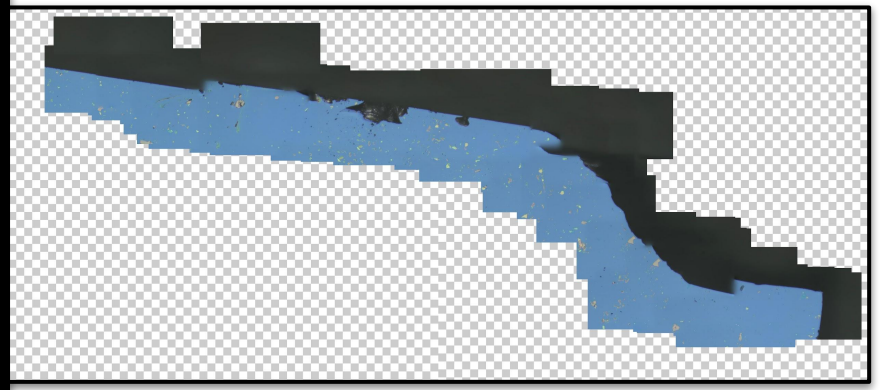
3. High resolution imaging from a large scale wafer



Camera software allows stitching the images together.

3. High resolution imaging from a large scale wafer





Example: Imaging SiO2/Si wafer

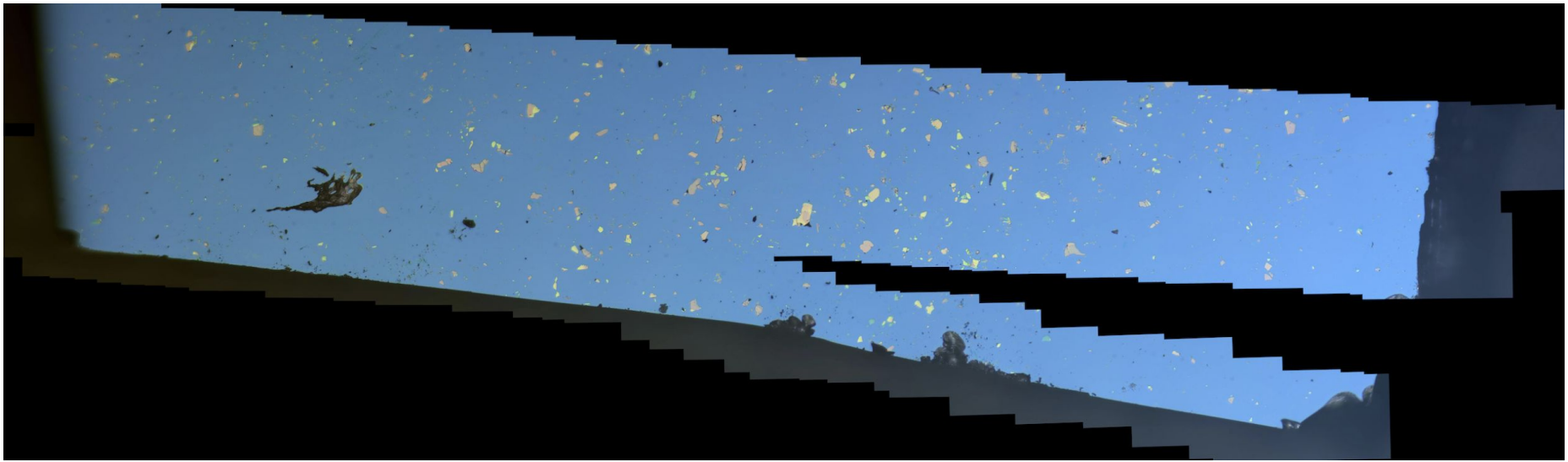
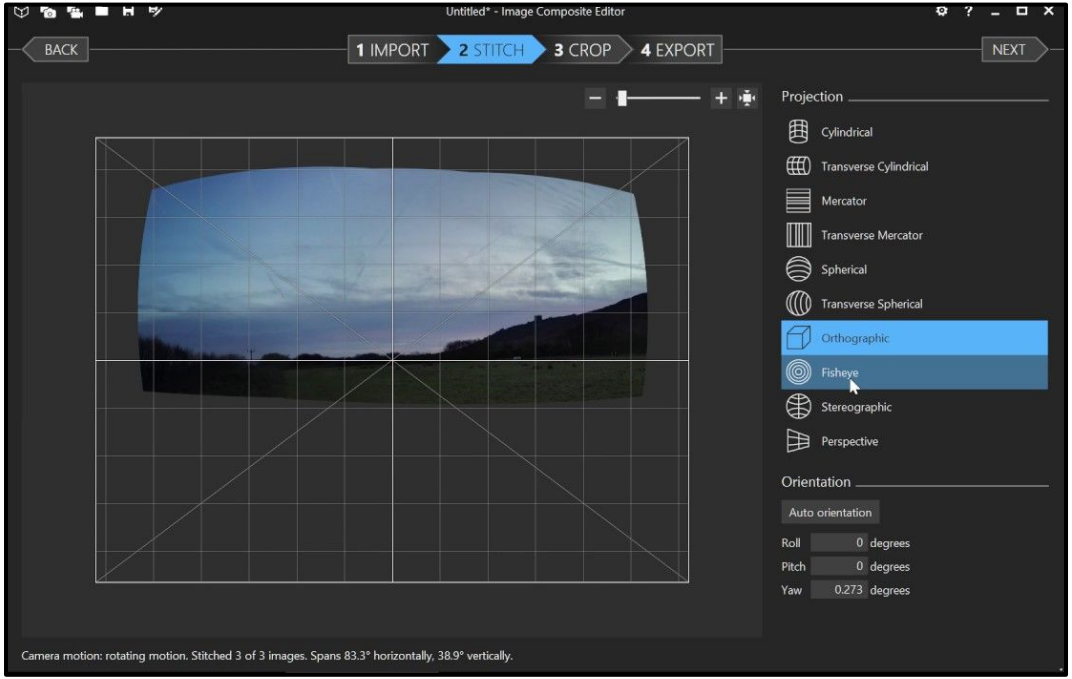
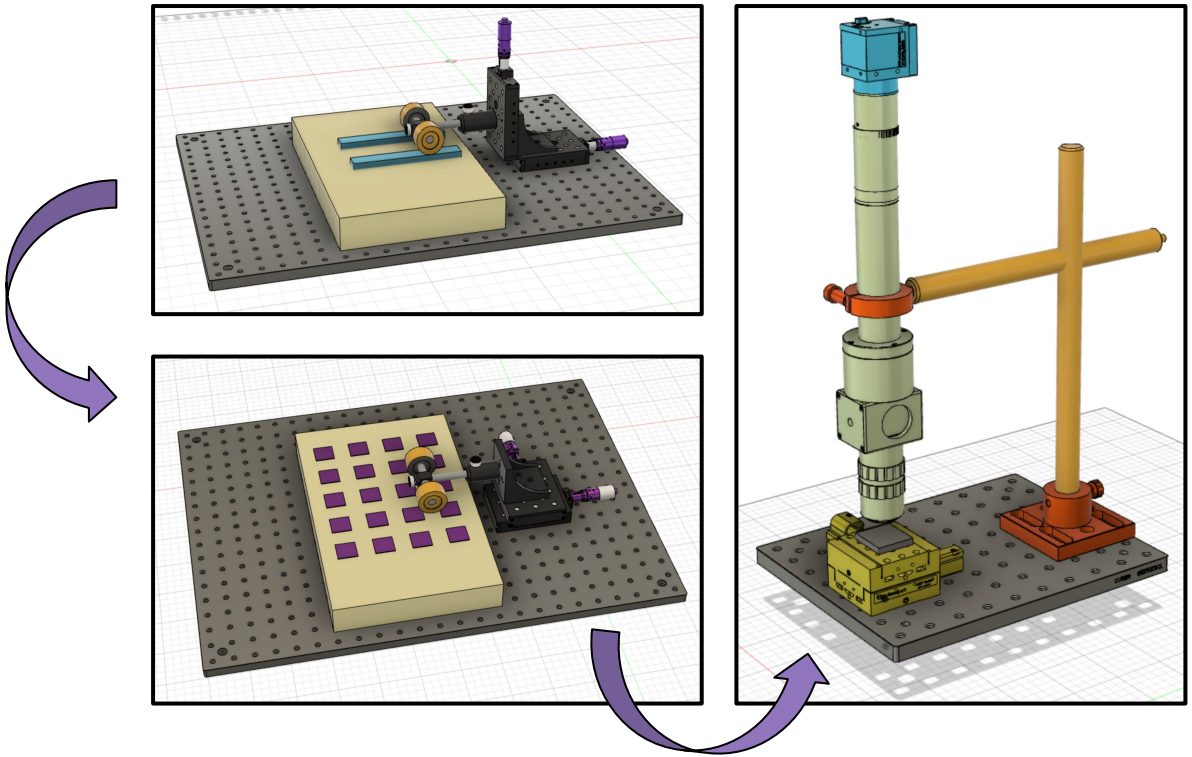


Image Composite Editor

- Alternative method
- Takes longer
- More accurate
- Does not need edges



Conclusion: How all 3 components work together

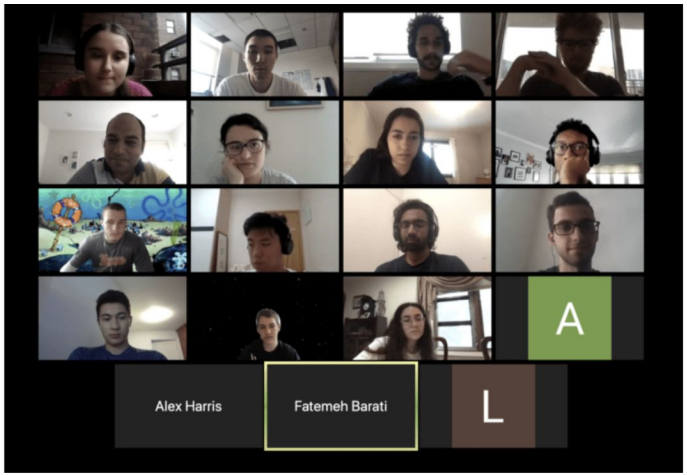


Thank You All Summer 2020 Q-Campers!

NYU

SHABANI LAB

QUANTUM MATERIALS & DEVICES



Quantum Guardians: Abigail Martucci, Eric Berg, Nikki Ebadollahi, Fatemeh Barati, and Prof. Javad Shabani

WAVEFUNCTION GENERATION AS A MEASURE FOR NOISE IN QUANTUM COMPUTING HARDWARE

QUANTUM TROOPERS:

Aurelia Brook, Daniel Li, Yuelong Li and Andreas Tsantilas

MOTIVATION

Feynman remarked that **the true power of quantum computers is in simulating quantum-mechanical systems**. This is believed to be intractable classically (in the class **NP**); however, we know that quantum simulation is in **BQP⊇P**, meaning that it is efficiently realizable on a quantum computer.

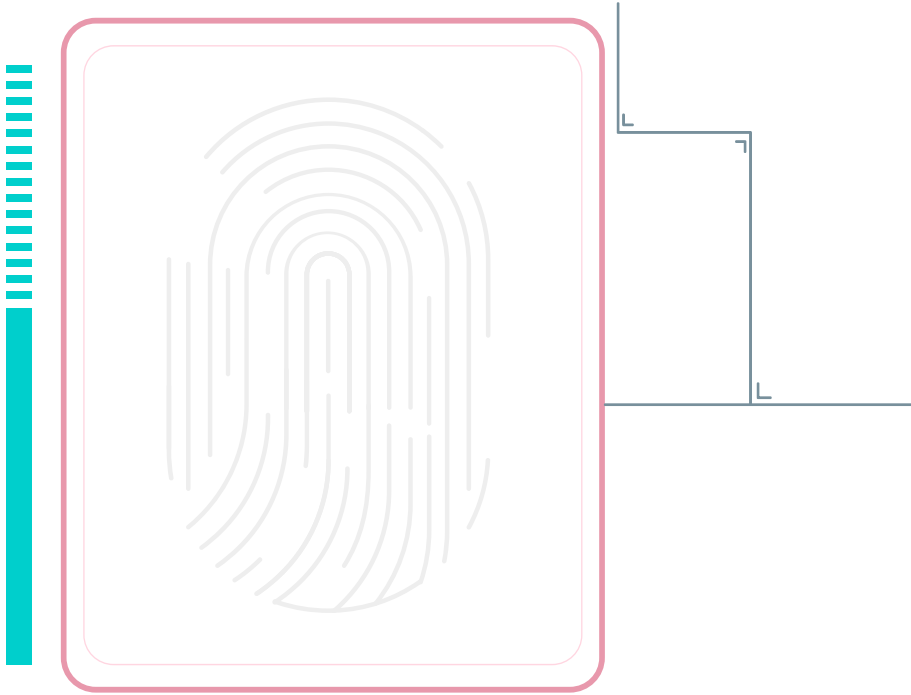
Preparing wavefunctions is a crucial first step in simulating a system; once the desired wavefunction is created, it can be evolved according to the laws of quantum mechanics.

Despite our increasing capability to control entanglement, we are still in the NISQ era, meaning such highly-entangled preparations are subject to large amounts of noise. **It is the focus of this project to characterize the effects of this noise on the Gaussian wavefunction, and we propose this as a metric for evaluating the noise on quantum hardware.**

Klco and Savage, arXiv:1904.10440 [quant-ph]

NOISY INTERMEDIATE-SCALE QUANTUM

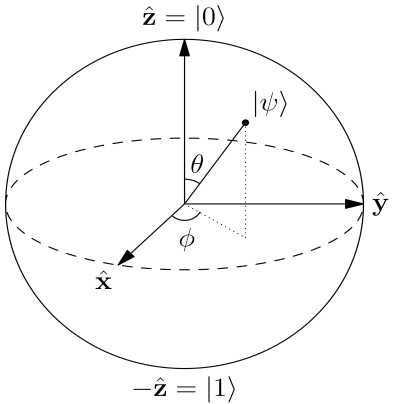
In order for quantum computers to be useful, we have to understand how noise impacts our system. Therefore, it's is useful to test how well a system can handle entanglement and other operations. This is a process known as benchmarking.



NOISE IN QUANTUM INFORMATION

BIT FLIP ERROR

- Θ rotation
- $\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|1\rangle + \beta|0\rangle$
- Characterized by operation elements $E_0 = \sqrt{p}\,\mathrm{I}, E_1 = \sqrt{1-p}\,\sigma_x$



PHASE ERROR

- Φ rotation
- $\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|0\rangle - \beta|1\rangle$
- Operation elements: $E_0 = \sqrt{p}\,\mathrm{I}, E_1 = \sqrt{1-p}\,\sigma_z = \sqrt{1-p}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

A single qubit state can be represented by a point on a sphere with the north and south poles as the |0⟩ and |1⟩ state.

Gates perform rotations between two quantum states.

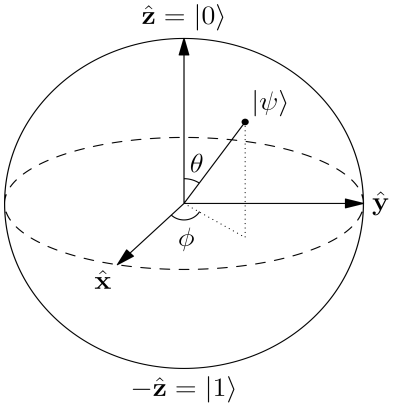
Noise for a state in Quantum information takes the form of unintended unitary transforms (rotations on the bloch sphere for a single qubit).

NOISE IN QUANTUM INFORMATION

AMPLITUDE DAMPING

- May cause bit reset
- Due to the dissipation of energy, state $|1\rangle$ may drop to $|0\rangle$ unexpectedly
- Operation elements:

$$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix},\ E_1 = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix}$$



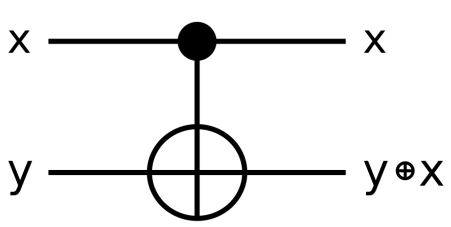
DEPOLARIZING ERROR

- The unexpected replacement of the original state by a completely mixed state $\frac{I}{2}$
- Under a probability p, (disregarding phase)

$$|\psi\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

GATE ERRORS

C-NOT



input		output	
x	y	x	y+x
|0⟩	|0⟩	|0⟩	|0⟩
|0⟩	|1⟩	|0⟩	|1⟩
|1⟩	|0⟩	|1⟩	|1⟩
|1⟩	|1⟩	|1⟩	|0⟩

Gate errors are defined by deviations from landing in the target state.

Different sources of errors such as depolarization or bit flip may act through gates and cause noise.

CNOT gates are large sources of error and notoriously hard to implement.

PREPARING GAUSSIAN WAVE FUNCTIONS ON QUBIT ARRAYS

The algorithm stated here prepares a Gaussian wavefunction on N qubits.

Implements an algorithm that iteratively prepares a discrete 1D Gaussian wave function on N qubits, with 2^N probabilities.

Lemma 1. The state of the periodic discrete Gaussian can be expressed recursively by

$$|\xi_{\mu,\sigma,N}\rangle = |\xi_{\frac{\mu}{2},\frac{\sigma}{2},N-1}\rangle \otimes \cos(\alpha)|0\rangle + |\xi_{\frac{\mu-1}{2},\frac{\sigma}{2},N-1}\rangle \otimes \sin(\alpha)|1\rangle$$

where the angle α is given by

$$\alpha = \cos^{-1}\left(\sqrt{\frac{g(\mu/2,\sigma/2)}{g(\mu,\sigma)}}\right)$$

Some hybridization with the more general Grover-Rudolph state preparation algorithm

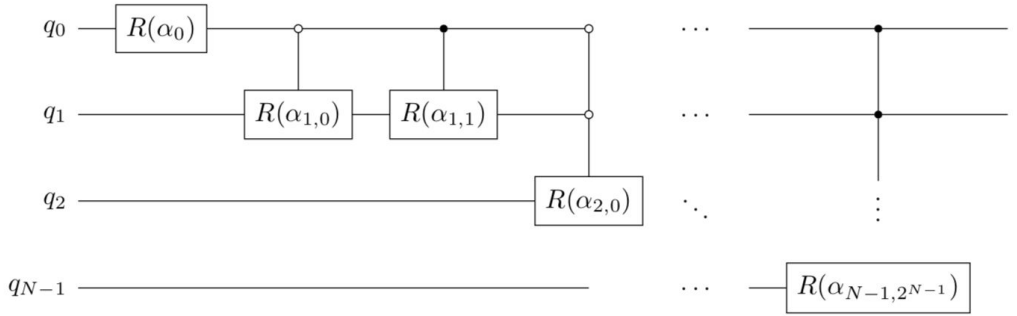
Algorithm 1 Kitaev-Webb

Input: The parameters $\mu, \sigma \in \mathbb{R}^+$, $k \in \mathbb{N}$, and $N \in \mathbb{N}$.
Output: A discrete approximation of the periodic state $|\xi_{\mu,\sigma,N}\rangle$.

1. **Initial State:** $|0^k\rangle|\mu,\sigma,N\rangle|0^N\rangle$;
2. **Compute** α in the first register, yielding the state $|\alpha\rangle|\mu,\sigma,N\rangle|0^N\rangle$;
3. **Apply the rotation** α **on** q_0, yielding the superposition state $|\alpha\rangle|\mu,\sigma,N\rangle|0^{N-1}\rangle \otimes (\cos(\alpha)|0\rangle + \sin(\alpha)|1\rangle)$
4. **Uncompute** α from the register containing it, obtaining $|0^k\rangle|\mu,\sigma,N\rangle|0^{N-1}\rangle \otimes (\cos(\alpha)|0\rangle + \sin(\alpha)|1\rangle)$
5. **Execute a change of parameters controlled by** q_0, given by $\cos(\alpha)|0^k\rangle|\frac{\mu}{2},\frac{\sigma}{2},N-1\rangle|0^{N-1}\rangle \otimes |0\rangle + \sin(\alpha)|0^k\rangle|\frac{\mu-1}{2},\frac{\sigma}{2},N-1\rangle|0^{N-1}\rangle \otimes |1\rangle$
6. **Whenever** $N > 1$**, Apply steps 2-5 on all qubits except the last**, giving $\cos(\alpha)|0^k\rangle|\frac{\mu}{2},\frac{\sigma}{2},N-1\rangle|\xi_{\frac{\mu}{2},\frac{\sigma}{2},N-1}\rangle \otimes |0\rangle + \sin(\alpha)|0^k\rangle|\frac{\mu-1}{2},\frac{\sigma}{2},N-1\rangle|\xi_{\frac{\mu}{2},\frac{\sigma}{2},N-1}\rangle \otimes |1\rangle$
7. **Reverse the parameter change**, yielding $|0^k\rangle|\mu,\sigma,N\rangle \otimes \left(\cos(\alpha)|\xi_{\frac{\mu}{2},\frac{\sigma}{2},N-1}\rangle|0\rangle + \sin(\alpha)|\xi_{\frac{\mu}{2},\frac{\sigma}{2},N-1}\rangle \otimes |1\rangle\right)$ $= |0^k\rangle|\mu,\sigma,N\rangle|\xi_{\mu,\sigma,N}\rangle$ as desired.

KITAEV-WEBB ALGORITHM

Adapting this to be more Qiskit-friendly

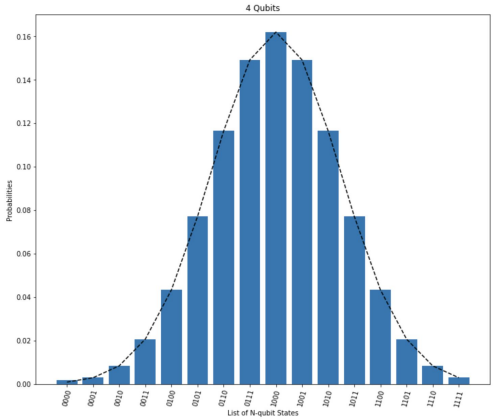


Notice how highly-entangled the circuit is. Because of this, goodness-of-fit to a Gaussian is a robust measure of how well one's hardware handles compounded noise.

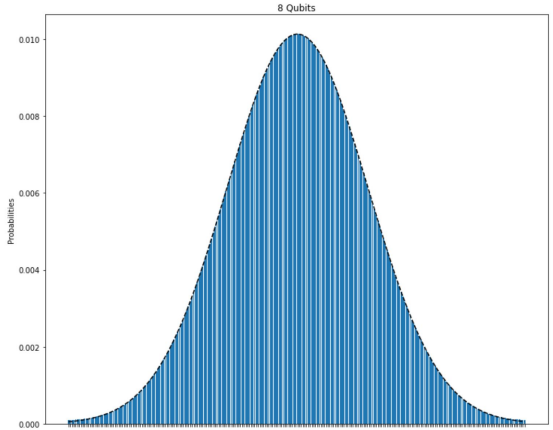
$$R(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$$

Kitaev and Webb, arXiv:0801.0342
Grover and Rudolph, arXiv:quant-ph/0208112

NOISELESS SIMULATIONS USING GAUSSIAN FIT



4 QUBITS

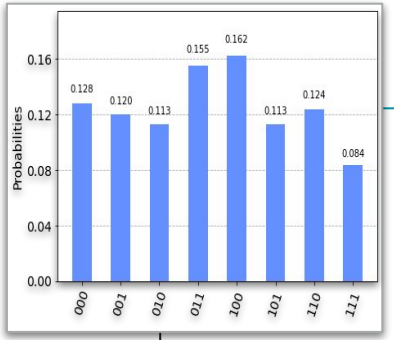


8 QUBITS

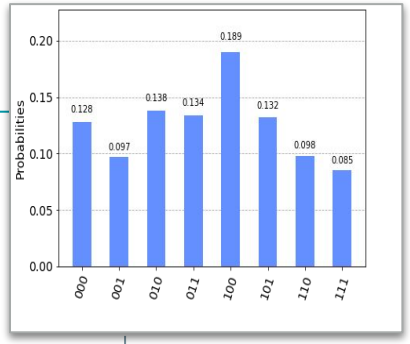
Pure simulation where the blue histogram shows binning of 100,000 simulated measurements against the dashed line of fitted gaussian

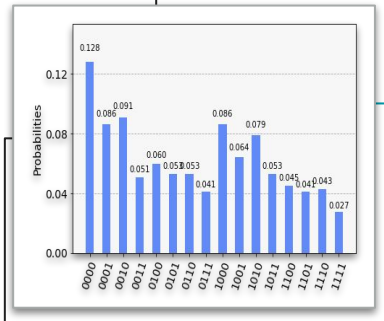
However, the state created is highly entangled and more subject to noisy deformations

GAUSSIAN FIT ON REAL DEVICES

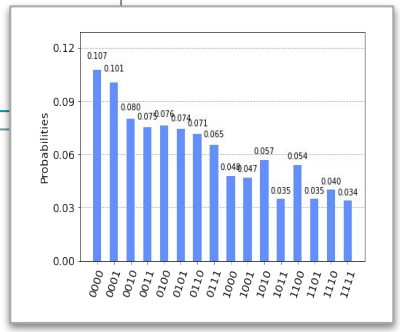


3 QUBITS





4 QUBITS



Run on ibmq_london

Simulated with ibm_london noise model on QASM

The left is run on the actual hardware, the right is simulated to perform like that hardware.

More qubits make the shape worse, due to the larger amount of CNOT gates.

EFFECTS OF NOISE ON REAL DEVICES

In order to understand the deviation from expected results when run on ibmq_london and other hardware, we simulated the same computation process.

qasm_simulator extracts noise models from the individual quantum devices.

When we coupled all the qubits, the gaussian computed looked better, or at least more symmetric.
The more qubits we used, the worse the actual and simulated results were; this is because there are more CNOTs used in higher-qubit calculations, so the results were noisier.

Evaluating Measuring Noise: Information Entropy

In order to measure noise, we proposed using a variant of what was used in a paper by Martinis et al. (2017) called CROSS ENTROPY DIFFERENCE.

$$H(X) = \sum_i q(x_i) \log q(x_i).$$

This is the equation for the Shannon entropy of a random variable. It can be taken as a measure of "uncertainty," or the information one gets after measuring a result.

$$H(p_{pcl}, p_U) \equiv -\sum_{x \in S} p_{pcl}(x_j|U) \log p_U(x_j).$$

For a sample of bit-strings S, we can find the cross entropy between a polynomial classical algorithm (probability p_pcl) and the ideal result of running a randomized circuit U

Entropy continued

Spoofing Linear Cross-Entropy Benchmarking in Shallow Quantum Circuits

Boaz Barak* Chi-Ning Chou† Xun Gao‡

May 7, 2020

Cross entropy has problems. Barak et al. (2020) showed that you can make a classical algorithm that gets better scores on cross entropy

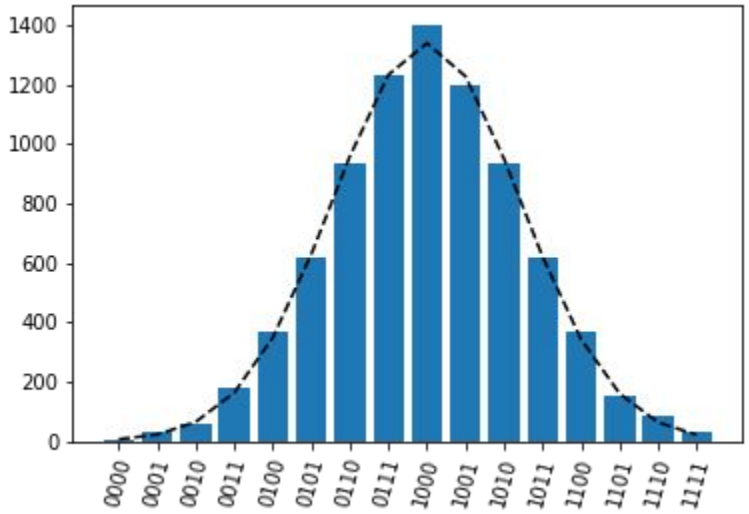
KL-Divergence (or relative entropy might be a better method at this time. D_KL = 0 iff p=q:

$$D_{KL}(p||q) \equiv \sum_i p(x_i) \log\left(\frac{p(x_i)}{q(x_i)}\right) = H(p,q) - H(p),$$

For a noisy implementation $\mathcal{U}$ of U, we can show the KL Divergence:

$$D_{KL}(\mathcal{U},U) = \sum_i |\langle x_i|\mathcal{U}|0\rangle|^2 \log\left(\frac{|\langle x_i|\mathcal{U}|0\rangle|^2}{|\langle x_i|U|0\rangle|^2}\right)$$

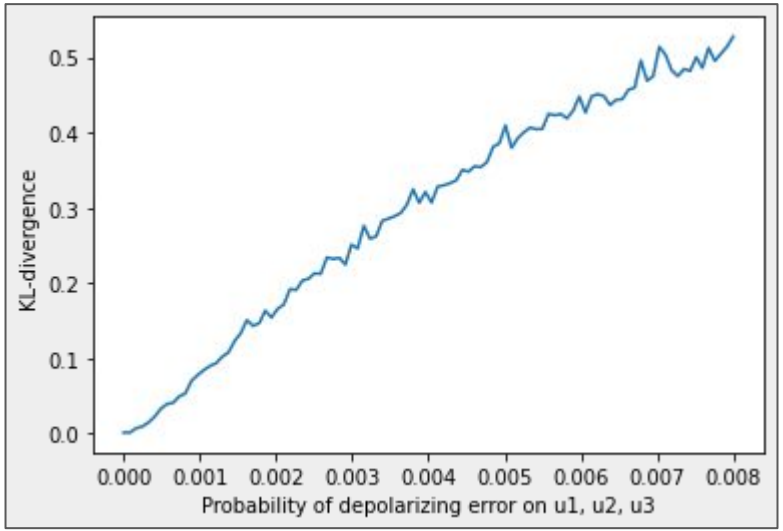
Relative Entropy of Ideal simulation



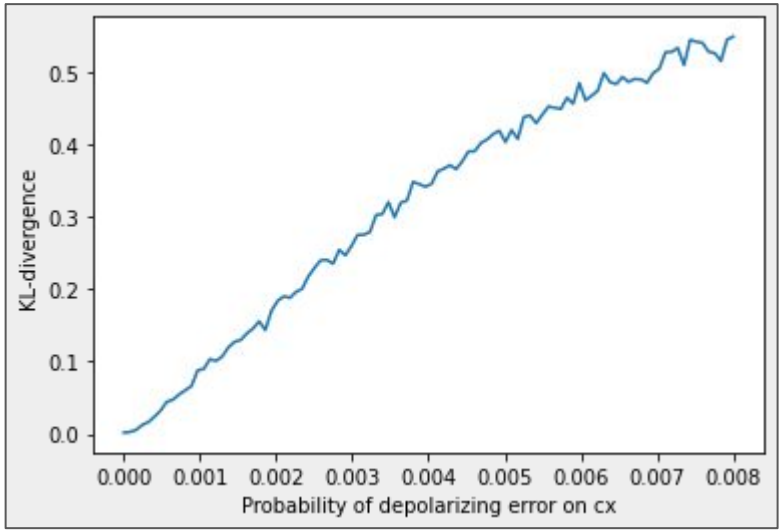
D_kl = 0.001426

KL-divergence of 4 q-bit Gaussian by simulated noises (QASM)

Depolarizing error, p=0~0.008, applied to u1, u2, u3

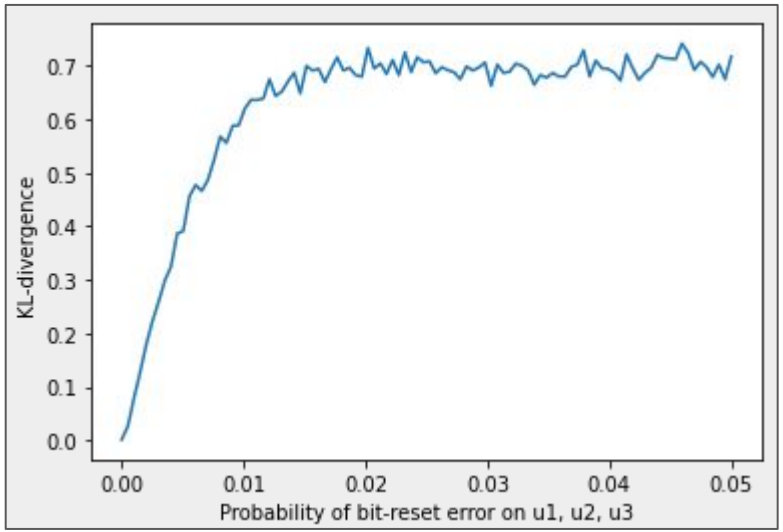


Depolarizing error, p=0~0.008, applied to cnot

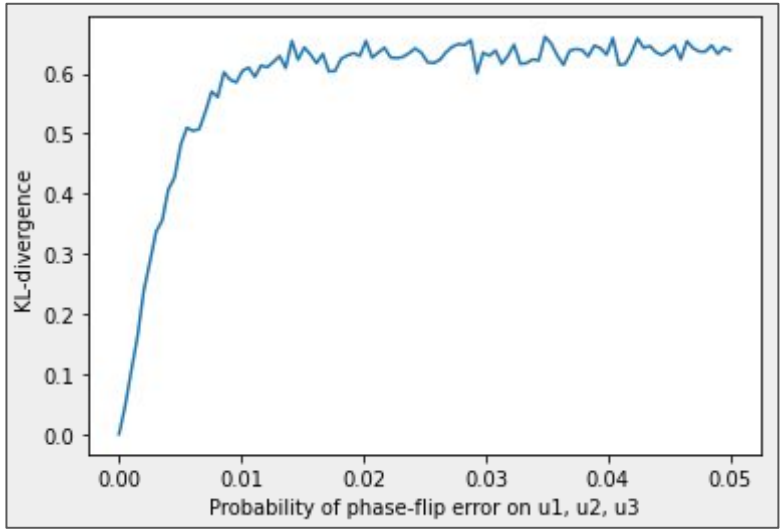


KL-divergence of 4 q-bit Gaussian by simulated noises (QASM)

Bit-reset error: p1=0~0.05, p2=0, applied to u1, u2, u3

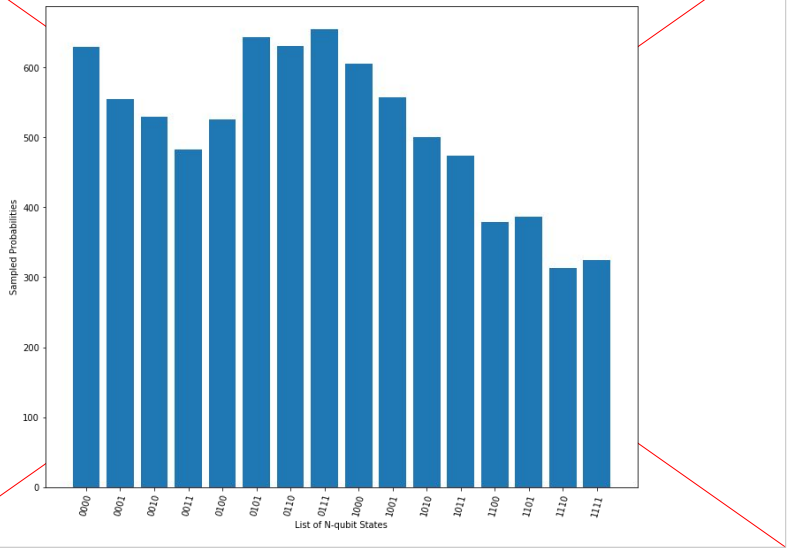


Phase-flip error: p=0~0.05, applied to u1, u2, u3

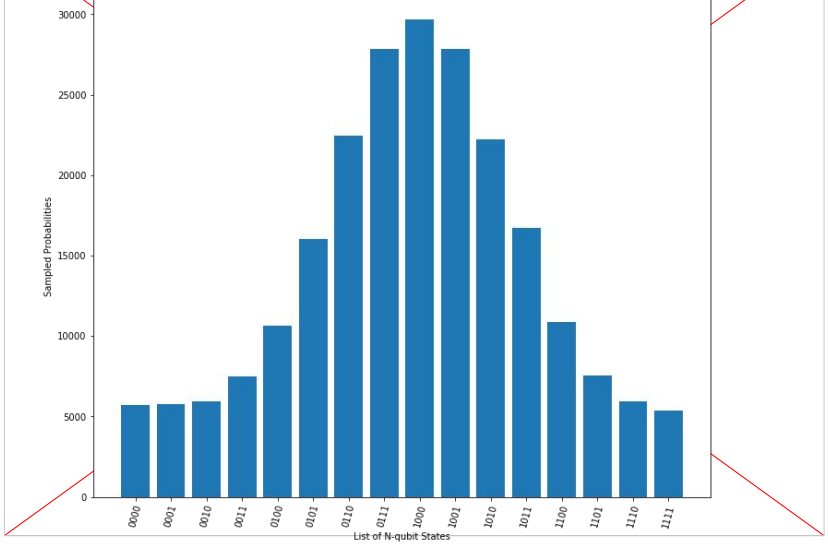


SIMULATED NORMAL DISTRIBUTION

Ibmq_rome simulation, perfect coupling, O(n^2).



Custom Simulation, error rates ~10^-3

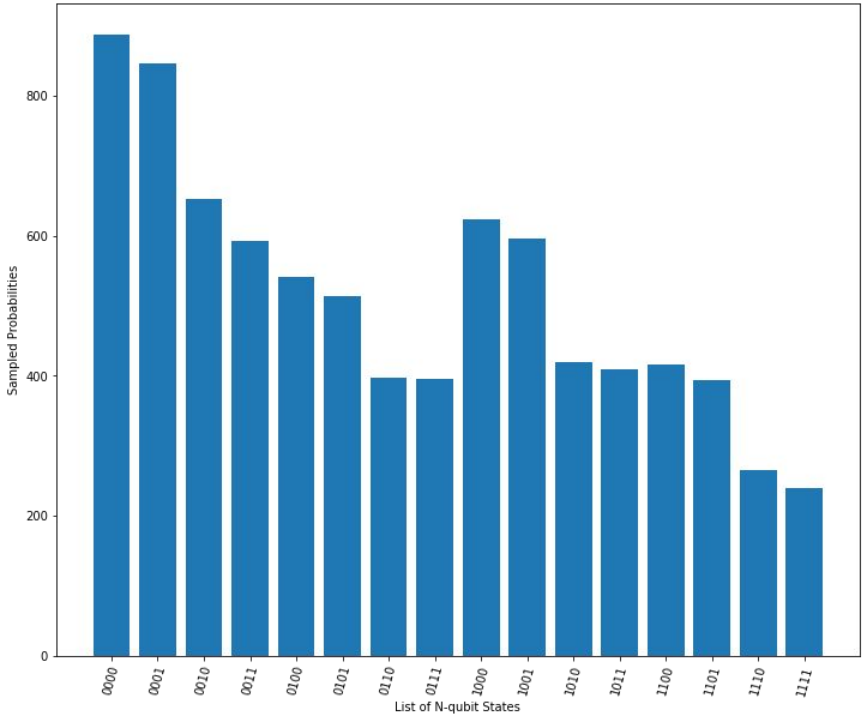


D_KL = 0.08413

FITTING TO NORMAL DISTRIBUTION ON REAL DEVICES

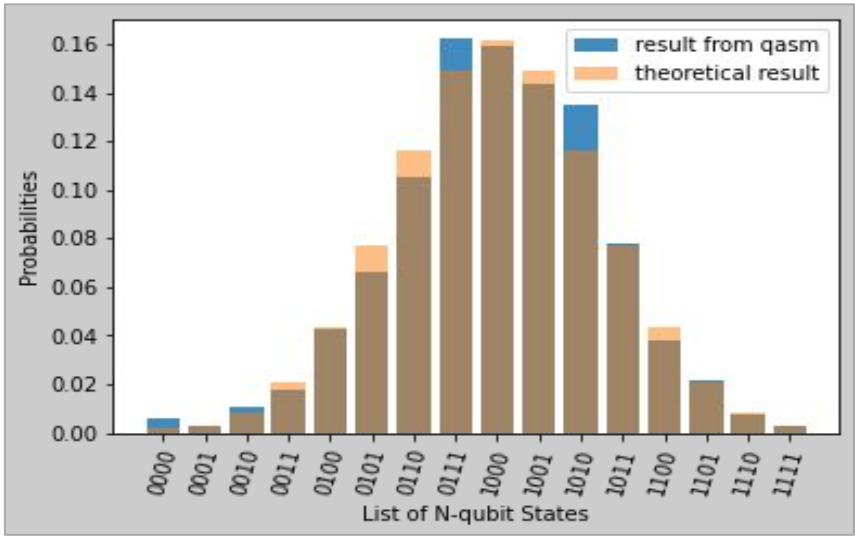
Run on ibmq_vigo, 4 qubits

D_KL = 0.5215



COMPUTATION AGAINST THEORY (past methods)

4 Qubit QASM simulation (1024 shots, noiseless)



Comparing to the expected gaussian calculated purely from theory:

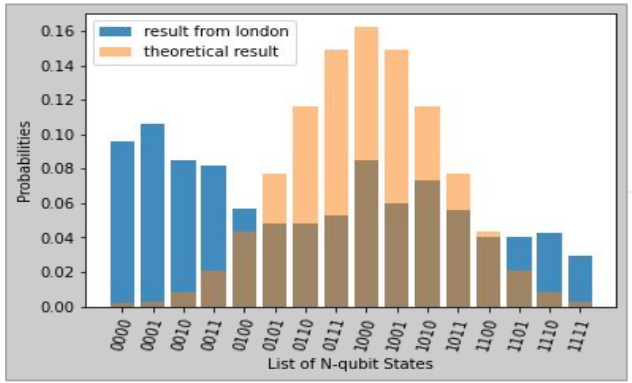
$$f(x,\sigma) = Ae^{\frac{-(x-x_0)^2}{2\sigma^2}}$$

Where x is the qubit state, $|x_0\rangle$ is the center state $|100\ldots\rangle$, A is a normalization factor, and sigma controls the width

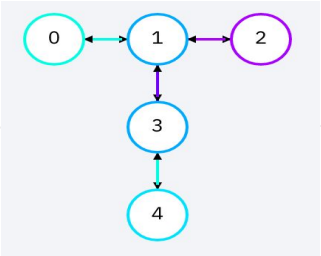
To the left we have a simulation on QASM

COMPUTATION AGAINST THEORY (past methods)

ibmq_london (1024 shots)

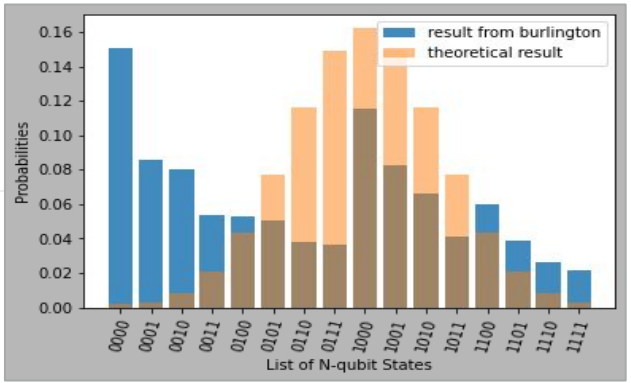


quadratic deviation: 3.93 x 10^{-3}



Coupling map

ibmq_burlington (1024 shots)



quadratic deviation: 4.16 x 10^{-3}

Two runs on real hardware against the expected gaussian showing a skew towards the |00...⟩ state

$$\text{quadratic deviation} = \sum_{i=1}^{N} \frac{(x_i - y_i)^2}{N}$$

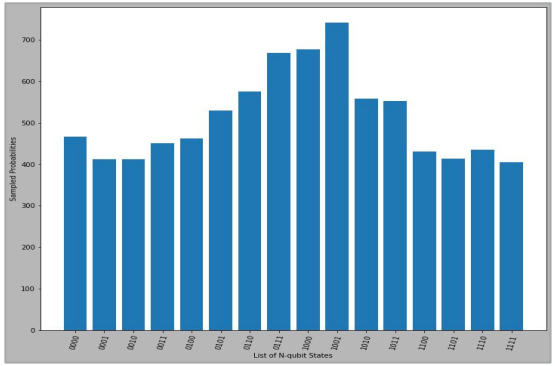
Where x_i/y_i are the data/fit

CONCLUDING REMARKS

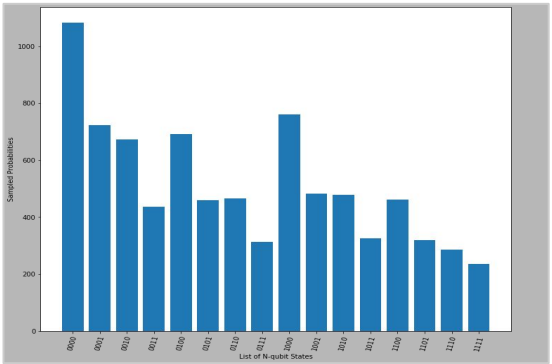
- We implemented a function to generate a highly entangled gaussian wave function on qiskit hardware
- Noisy simulations and chips result in skewed distributions, which seem configuration-dependent, as well as affected by N.
    - N > 3 is when the asymmetry becomes manifestly compounded by the number of CNOTs in the circuit

NOISE IN STATE GENERATION

- We began isolating <u>how certain types of noise distort the Gaussian.</u>
- For the top graph, this is a custom QASM simulation with only depolarizing error. We observe this error resulting in a broadened gaussian. This is because with probability p, the maximally entropic state is returned (Identity/2).
- For the bottom graph, this is a custom QASM simulation with only bit-reset errors. We observe a leftward skew in the simulation with higher probability to default to $|0\rangle$, due to the energy required to keep the qubits at the 1 state.



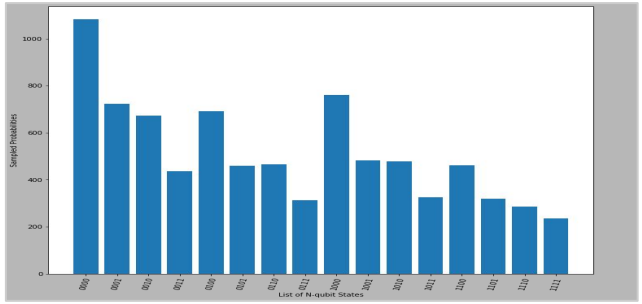
quadratic deviation: 8.21*10^-3



quadratic deviation: 1.72*10^-2

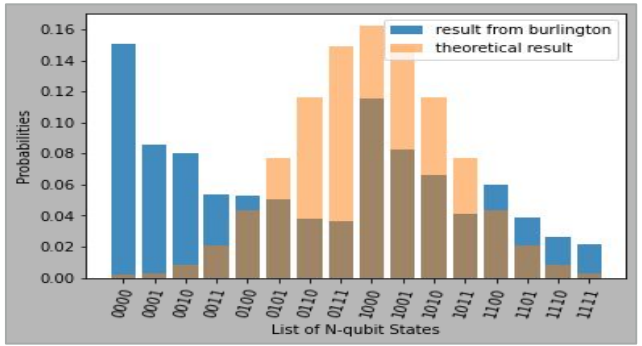
NEXT STEPS

QASM bit-reset simulation



quadratic deviation: 1.72*10^-2

ibm_burlington



- The behavior of the QASM simulation under bit-reset errors seems to be more dominant than the broadening seen from depolarization in the results we find for ibm_burlington for example.
- If we know the sources of error, we can obtain better calculations [1] and extend the power of noisy real hardware.
- We also began exploring methods of error-correcting codes such as repetition cotes, in order to mitigate the noise.

[1] Kandala, A., Temme, K., Córcoles, A.D. et al. Error mitigation extends the computational reach of a noisy quantum processor.

ACKNOWLEDGEMENTS

Thank you to all who helped and participated in the Summer Quantum Camp 2020 for an unforgettable experience

Special thank you to Professor Shabani, Joseph Yuan, and Bassel Elfeky

QUESTIONS?

REFERENCES

- Klco, Natalie, and Martin J. Savage. “Minimally Entangled State Preparation of Localized Wave Functions on Quantum Computers.” Physical Review A 102.1 (2020): n. pag. Crossref. Web.
- Grover, Lov, and Terry Rudolph. "Creating superpositions that correspond to efficiently integrable probability distributions." *arXiv preprint quant-ph/0208112* (2002).
- Kitaev, Alexei, and William A. Webb. "Wavefunction preparation and resampling using a quantum computer." *arXiv preprint arXiv:0801.0342* (2008).
- Kandala, A., Temme, K., Córcoles, A.D. et al. Error mitigation extends the computational reach of a noisy quantum processor. Nature 567, 491–495 (2019). https://doi.org/10.1038/s41586-019-1040-7
- Boixo, Sergio, et al. "Characterizing quantum supremacy in near-term devices." *Nature Physics* 14.6 (2018): 595-600.
- Barak, Boaz, Chi-Ning Chou, and Xun Gao. "Spoofing Linear Cross-Entropy Benchmarking in Shallow Quantum Circuits." *arXiv preprint arXiv:2005.02421* (2020).