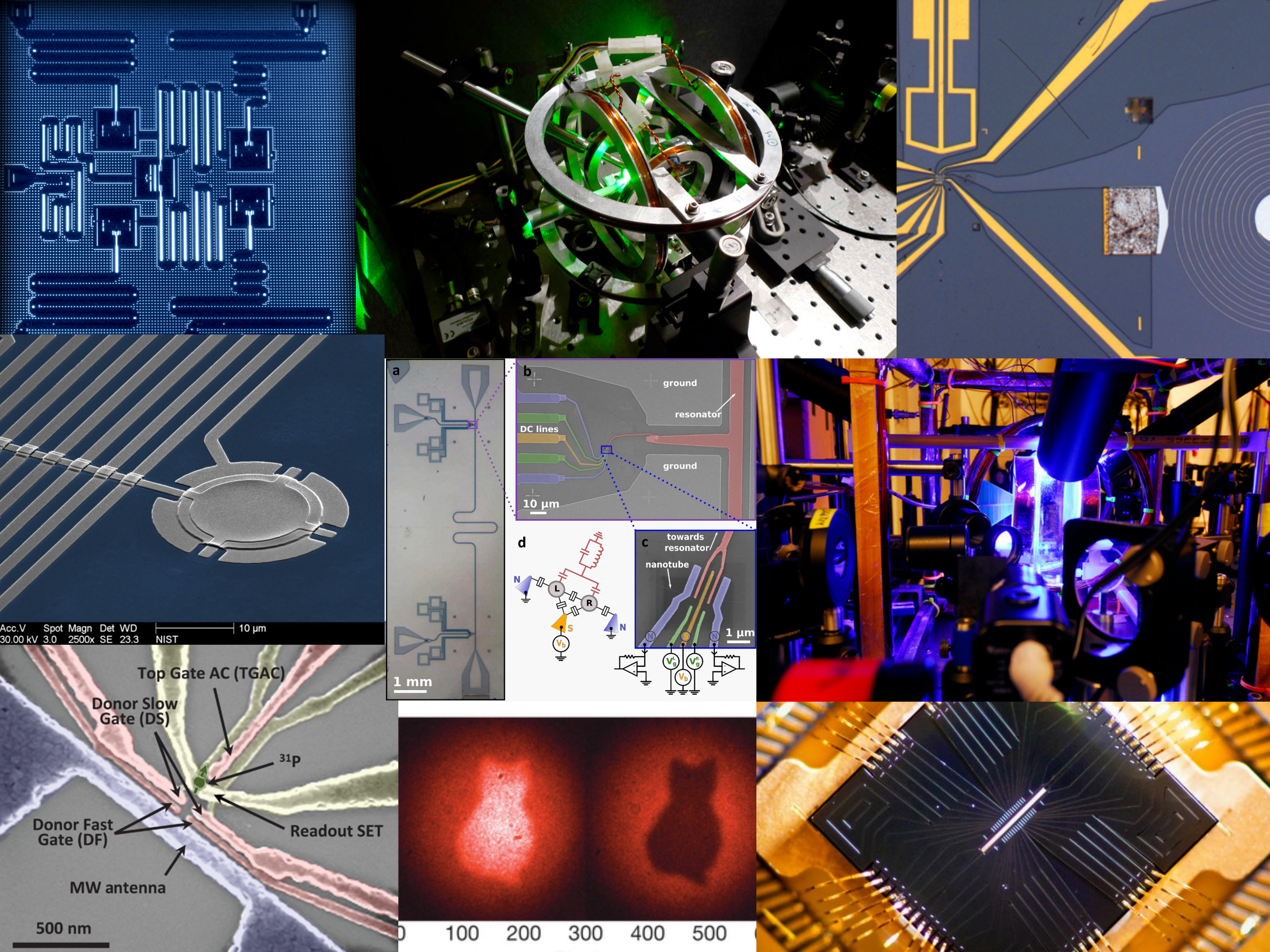
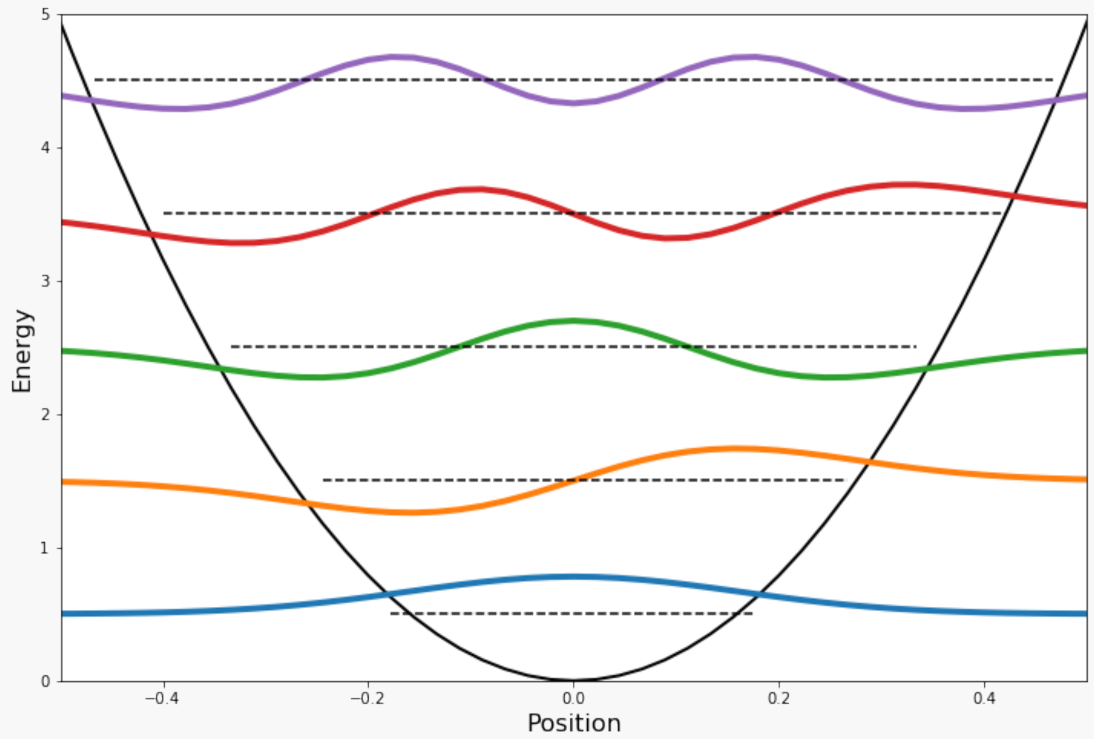
Simulating Quantum Optical Systems on Classical Computers

Paul Nation
QuSTaR

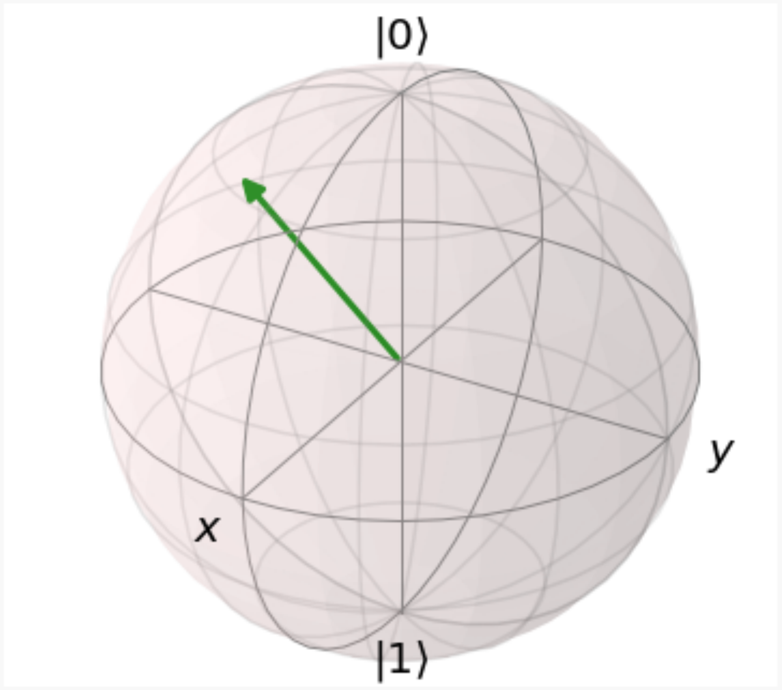

Acc.V Spot Magn Det WD
10 μm
30.00 kV 3.0 2500x SE 23.3
NIST
a
b
ground
resonator
DC lines
ground
10 μm
1 mm
d
N
L
R
S
V_b
N
c
towards resonator
nanotube
1 μm
Top Gate AC (TGAC)
Donor Slow Gate (DS)
^{31}P
Donor Fast Gate (DF)
Readout SET
MW antenna
500 nm
100
200
300
400
500

What do all of these systems have in common?

1) Can all be described using two basic building blocks:



Oscillators
(harmonic, nonlinear)



Spins
(qubits, qudits)

Quantum optical systems

2) All interesting systems are nonlinear; likely need to do computer simulations.

Problem: Resources needed to simulate classically scale exponentially.

$$|\Psi\rangle = a|\uparrow\rangle + b|\downarrow\rangle, \quad a, b \in \mathbb{C}$$ Single qubit

$$|\Psi\rangle = a|\uparrow\uparrow\rangle + b|\downarrow\uparrow\rangle + c|\uparrow\downarrow\rangle + d|\downarrow\downarrow\rangle$$ Two qubits

N qubits requires 2^N complex numbers to specify.

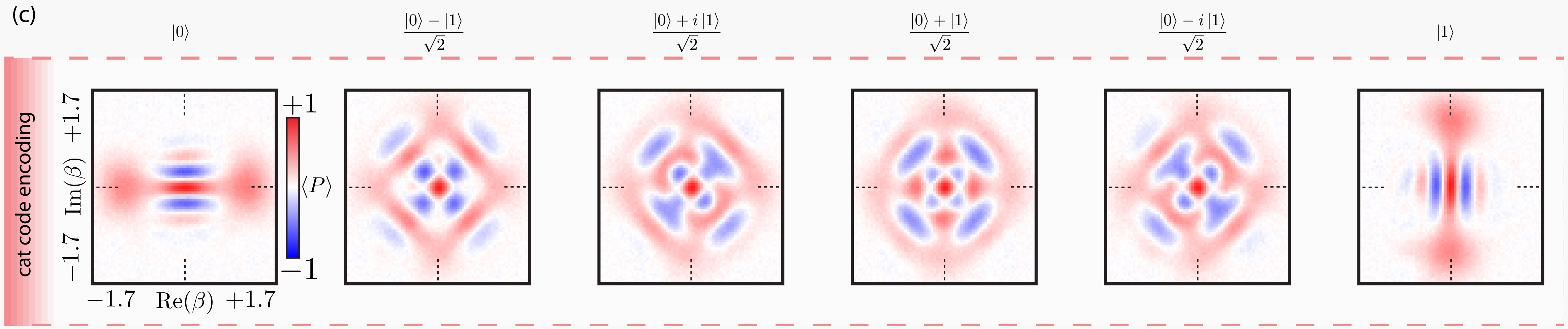
Feynman: Simulate quantum systems with a quantum computer.

Currently big business!

Near term quantum computers of limited applicability.

Technology comes to the rescue.

Generation and manipulation of quantum systems at the few or single excitation level.



[Ofek et al., Nature **536**, 441 (2016).]

Systems live in a truncated Hilbert space.

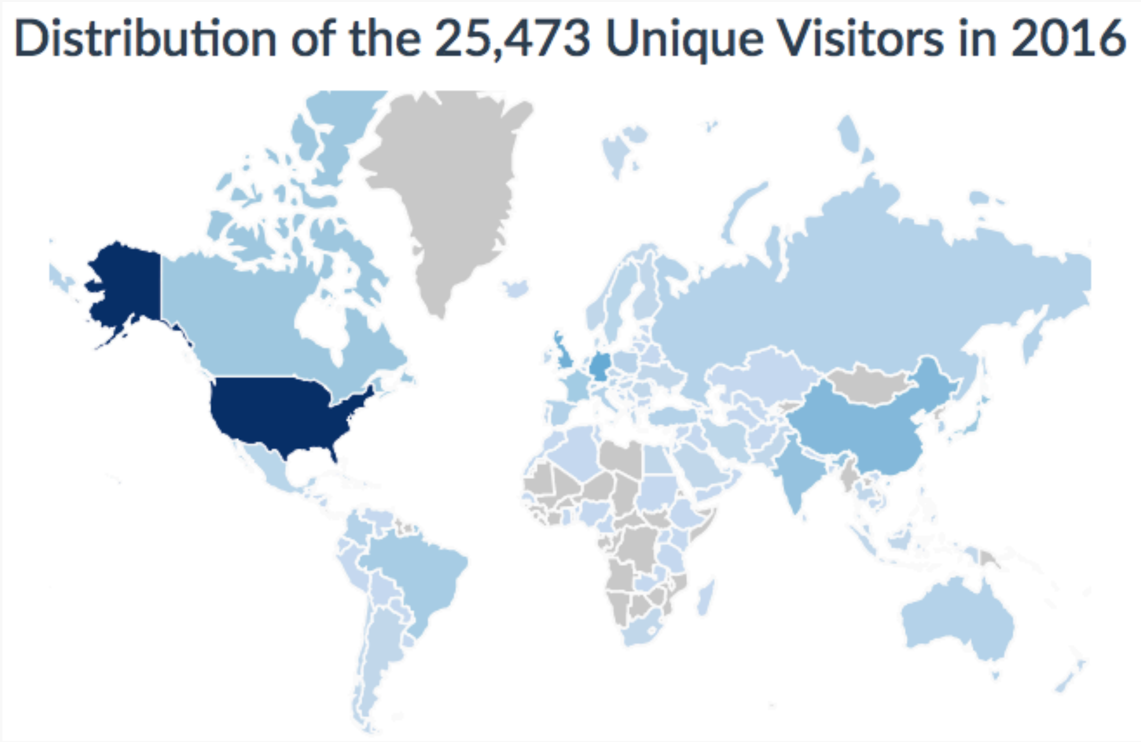
Truncated Hilbert space + Few system components = Classically tractable simulations

QuTiP: Quantum Toolbox in Python

Open-source software for simulating quantum optical systems.

Started at Dartmouth in 2010.

Used by nearly every major university, research lab, or commercial entity interested in quantum optical systems.



Very easy to learn; write code like the mathematics.

Large community of helpful users, and actively developed.

qutip.org

Very easy to install:

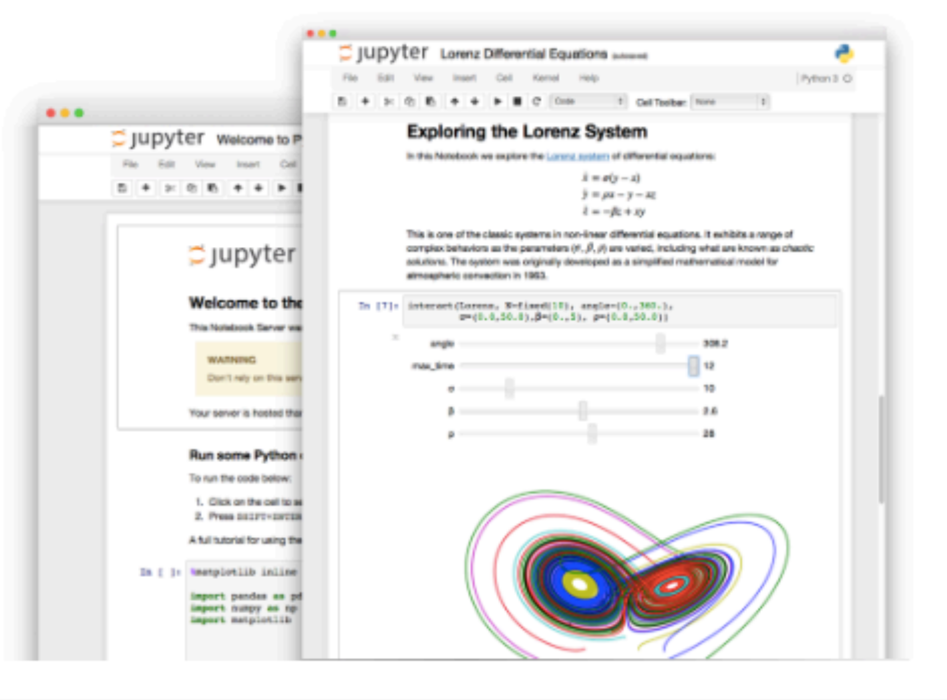
- Get Anaconda Python: continuum.io/downloads
- Run in command line:

```
>> conda config --append channels conda-forge
>> conda install qutip
```

A quick intro. to Python itself is at: qutip.org/tutorials

Lecture stuff: github.com/nonhermitian/dartmouth_2017

We will be using the Jupyter notebook interface.



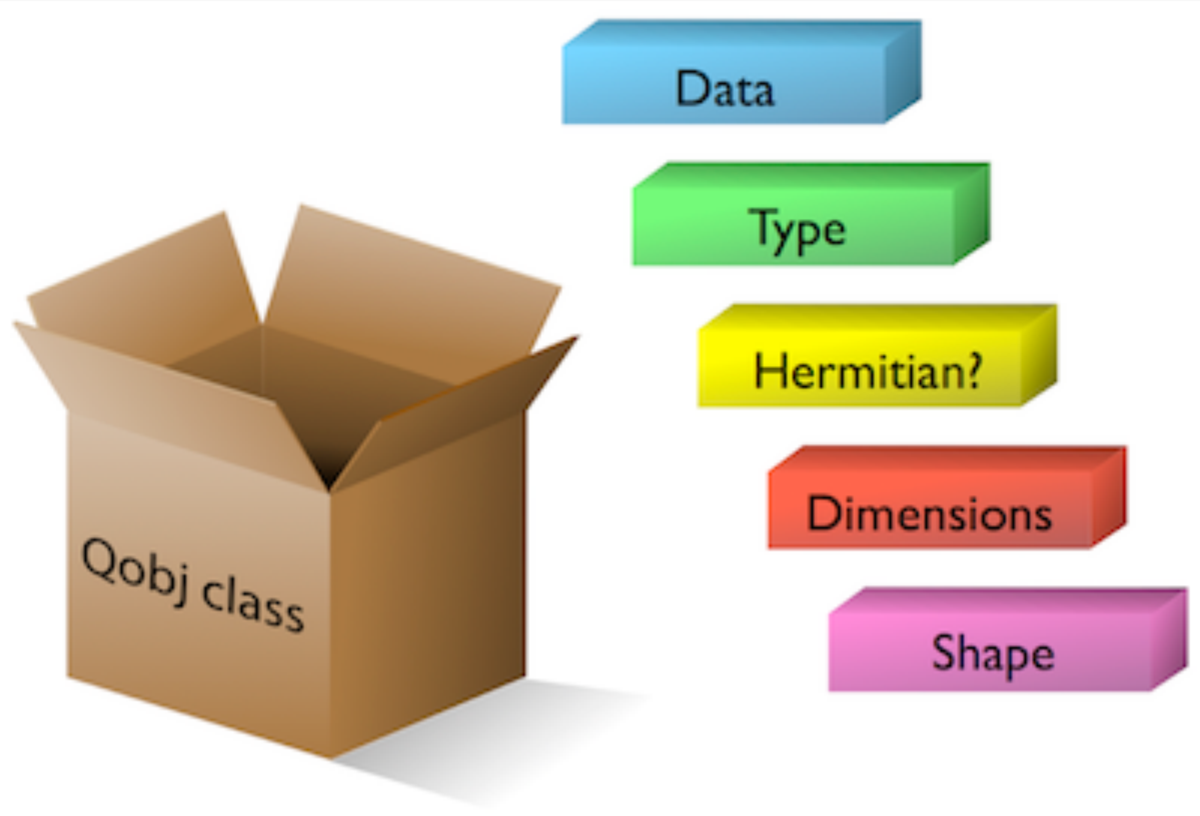
Constructing Operators and States

(Part 1/2)

The Qobj Class: The fundamental object in QuTiP

The Qobj class is a container whose contents (attributes) specify a given quantum state or operator.

The matrix representation (data) is stored as a sparse array.



Attributes can be accessed: `Q.data`,`Q.dims`,…

Classes also have `methods' that perform actions:

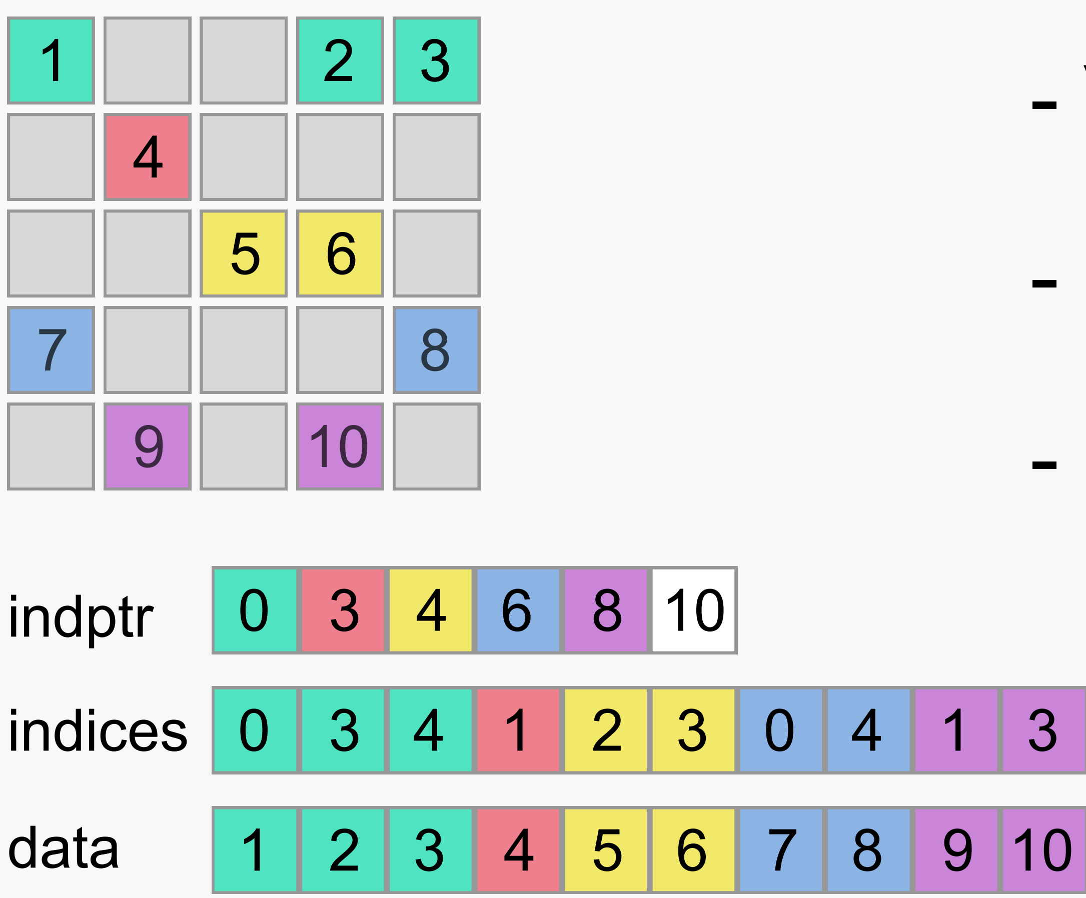
```
Q.unit() # Normalize          Q.ptrace(…) # Partial trace
Q.dag()  # Adjoint            Q.expm()    # Matrix exp.
Q.norm() # Norm               Q.full(…)   # Return dense matrix
Q.eigenstates(…)              Q.tr(…)     # Trace
```

Sparse Matrices in QuTiP

Most physically relevant operators and states are sparse in a correct choice of representation.

The density of nonzero elements rapidly decreases with Hilbert space size.

QuTiP uses the CSR (compressed sparse row) format.



- Very fast row operations.
- Fast linear algebra.
- Widely used. (e.g. Mathematica)

The most important sparse operation is sparse-matrix dense-vector multiplication (SpMV):

1			2	3
	4			
		5	6	
7				8
	9		10	

```
for row in range(num_rows):
    tot = 0
    for idx in range(indptr[row], indptr[row+1]):
        tot += data[idx] * vec[indices[idx]]
    out_vec[row] = tot
```

indptr	0	3	4	6	8	10				
indices	0	3	4	1	2	3	0	4	1	3
data	1	2	3	4	5	6	7	8	9	10

Used everywhere: time-evolution, eigensolving, exponentiation, expectation values, ...

This, like most sparse operations, scales with the number of nonzero elements (NNZ) in the matrix.

Constructing Operators and States

(Part 2/2)

Composite Systems

Typical systems of interest are made up of a collection of fundamental components.

Need an operation that combines two or more Hilbert spaces together.

Given systems A and B, the combined system is given by the tensor product: $A \otimes B$.

Example $\sigma_z \otimes \mathbb{I}_2$:

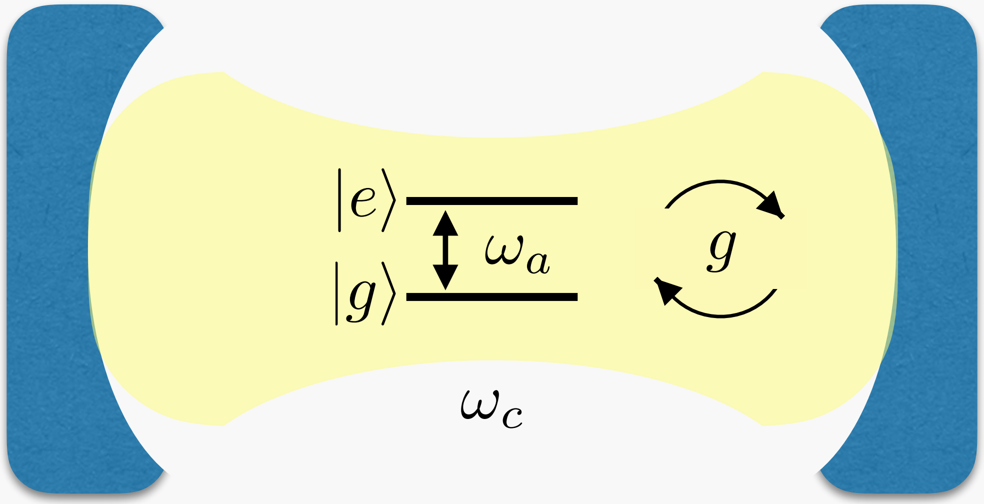
$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 0 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & -1 \begin{bmatrix} 1 & 0 \\ 0 & 1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

In QuTiP, this operation is called `tensor`:

```
qt.tensor(qt.sigmaz(),qt.identity(2))
```

Operators operating on a single subsystem get tensored with the identity acting on the other subspace(s).

Example: Jaynes-Cummings Model



If $g \ll \omega_c, \omega_a$ (RWA):

$$H = \hbar\omega_c a^\dagger a + \hbar\omega_a \sigma_z + \hbar g(a^\dagger \sigma_- + a\sigma_-^\dagger)$$

Jaynes-Cummings Model

(Part 1/3)

Time Evolution

For a closed, time-independent system, dynamics given by Schrodinger equation ($\hbar = 1$):

$$\frac{d|\psi(t)\rangle}{dt} = -iH|\psi(t)\rangle \quad \rightarrow \quad |\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$$

Recall that $e^A = \sum_n A^n/n!$.

Unless H is diagonal, idempotent, propagator is dense.

Never explicitly form the propagator unless really need it!

Solve original ODE problem.

Involves repeated use of SpMV.

In QuTiP, Schrodinger evolution performed by:

```
qt.sesolve(H, psi0, tlist, e_ops, args, options)
```

`H` = Hamiltonian

`psi0` = initial state vector (or density matrix)

`tlist` = times over which to evaluate dynamics

`e_ops` = operators for computing expectation values

Output of all dynamics solvers is returned in a Result class.

Jaynes-Cummings Model

(Part 2/3)

Partial Trace

The output state vectors are for the combined system.

How do we extract the state of just the cavity (atom) as a function of time?

Density matrix of system A returned by partial trace over system B.

$$\rho_A \equiv \mathrm{Tr}_B(\rho_{A\otimes B})$$

$$\mathrm{Tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = |a_1\rangle\langle a_2|\mathrm{Tr}(|b_1\rangle\langle b_2|)$$

This is the unique operation that gives the correct measurement statistics for subsystems.

Given a pure input state for the full system, subsystems themselves need not be pure.

Ex: $|\psi_{A\otimes B}\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right)$

$$\rho_A = \frac{1}{2}\left[\mathrm{Tr}_B(|00\rangle\langle 00|) + \mathrm{Tr}_B(|11\rangle\langle 00|) + \mathrm{Tr}_B(|00\rangle\langle 11|) + \mathrm{Tr}_B(|11\rangle\langle 11|)\right)]$$

$$\rho_A = \frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|) = \frac{\mathbb{I}}{2}$$

Subsystem is in maximally mixed state; traced over classical and quantum correlations between subsystems.

For bipartite system, this can be used as entanglement measure via Von-Neumann entropy

$$S(\rho_A) = -\mathrm{Tr}(\rho_A \log \rho_A)$$

Jaynes-Cummings Model

(Part 3/3)

Wigner Function

Quantum quasi-probability distribution

$$W(x,p) = \frac{1}{\pi\hbar}\int_{-\infty}^{\infty} \langle x+y|\rho|x-y\rangle e^{-2ipy/\hbar} dy$$

has the following properties:

$$\int_{-\infty}^{\infty} dp\; W(x,p) = \langle x|\rho|x\rangle \qquad \int_{-\infty}^{\infty} dx\; W(x,p) = \langle p|\rho|p\rangle$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dx dp\; W(x,p) = \mathrm{Tr}(\rho) = 1$$

Knowing the Wigner function, is equivalent to knowing the density matrix.

Wigner function can be negative.

Only some nonclassical states can exhibit negative Wigner functions; allowed since $[x, p] \neq 0$.

Visualization of the Wigner function can be used to verify genuine quantum states.