Make your code count
Quantum simulations and collaborative code

Shahnawaz Ahmed

QuTiP: The Quantum Toolbox in Python
## About me

<table>
<thead>
<tr>
<th>Year</th>
<th>Details</th>
<th>Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019</td>
<td>PhD @ Applied Quantum Physics, Chalmers University of Technology, Sweden</td>
<td>Quantum simulations, Machine Learning</td>
</tr>
<tr>
<td></td>
<td>Anton Fisk Kockum, Prof. Göran Johansson</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>Master’s thesis @ Theoretical Quantum Physics Lab, Riken, Japan</td>
<td>Spin-boson model Ultrastrong coupling</td>
</tr>
<tr>
<td></td>
<td>Mauro Cirio, Neill Lambert, Prof. Franco Nori</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>Intern @ Theoretical Quantum Physics Lab, Riken, Japan</td>
<td>Deep Learning</td>
</tr>
<tr>
<td></td>
<td>Nathan Shammah, Clemens Gneiting, Prof. Franco Nori</td>
<td>Collective effects in large spin systems</td>
</tr>
<tr>
<td>2016</td>
<td>Intern @ Google Summer of Code</td>
<td>Diffusion Imaging</td>
</tr>
<tr>
<td></td>
<td>Ariel Rokem, Eric Peterson, Rafael Henriques</td>
<td></td>
</tr>
</tbody>
</table>
Do you guys put “Quantum” in everything?

- Ant-man and the Wasp (2018)

- Quantum Machine Learning
- Quantum Big Data
- Quantum Neural Networks
- Quantum Cryptography
- Quantum Sensing (GPS)
- Quantum Internet
Interests @ FOSDEM19

- Photonic quantum computing
- Standardization and availability of code
- Optimization and control
- NISQ - Noisy intermediate scale quantum
- Spin systems, quantum annealing
- Hybrid quantum classical algorithms
- Quantum circuits
- Cloud quantum computing
Quantum physics simulator

- Photonic quantum computing
- Standardization and availability of code
- Optimization and control
- NISQ - Noisy intermediate scale quantum
- Quantum circuits
- Cloud quantum computing
- Hybrid quantum classical algorithms
- Spin systems, quantum annealing
Quantum physics simulator

A collaborative effort over many years by the community

- cQED
- Quantum optics
- Ion Traps
- Superconducting circuits
- Condensed matter
- Error correction
- Optomechanics

QuTiP

and more …
The Quantum Toolbox in Python

**Built with Python**
Python's straightforward syntax allows for constructing, manipulating, and evolving quantum objects using QuTiP with just a few lines of code. QuTiP is the ideal toolbox for research or the classroom.

**Custom algorithms**
QuTiP can determine if an operator is Hermitian without performing the conjugate transpose. This is just one of many custom algorithms devised to maximize performance. Sparse matrices deployment efficiently manipulates large datasets.

**Built-in solvers**
A variety of built-in solvers allow the study of dynamical simulations and steady-state analysis. In addition to Lindblad and Monte Carlo solvers, QuTiP offers advanced routines for Bloch-Redfield and Floquet formalism, and non-Markovian systems.

**Fast**
QuTiP is capable of leveraging the multiprocessing power inside every modern computer. QuTiP can take advantage of the Python multiprocessing library, OPENMP, SSE3 processor extensions, and Intel MKL.

**C++ performance**
A wide range of time-dependent evolution simulations can be runtime compiled into C++ behind the scenes using Cython. The ease of use of Python is boosted by compiled code.

**Experimental Data**
If you need to construct a function from a data set, QuTiP allows for passing interpolating functions as time-dependent arguments to the evolution solvers, also runtime compiling into C++.

**Ad-hoc visualization tools**
From Bloch spheres to nonlinear colormaps for Wigner functions, QuTiP includes a host of built-in visualization routines that help bring data to life, including through animations and 3D graphics.

**Independent testing**
QuTiP is thoroughly tested, both by its thousands of users, and by a large collection of built-in test scripts independently run by Travis CI. Over a thousand such tests help cover nearly all of the built-in functions, continuously running in development.

**User friendly**
No software should be a black box to the user, especially in science. QuTiP includes hundreds of pages of documentation, a multitude of tutorial Jupyter notebooks, and a friendly community of users who help answer questions.
### Open source quantum (2016 -)

<table>
<thead>
<tr>
<th>Year</th>
<th>Project</th>
<th>Language</th>
<th>Team/Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>QETLAB</td>
<td>Matlab</td>
<td>University of Waterloo, Canada</td>
</tr>
<tr>
<td>2016</td>
<td>Liqui&gt;</td>
<td>F#</td>
<td>Microsoft</td>
</tr>
<tr>
<td>2016</td>
<td>Quantum Fog</td>
<td>Python</td>
<td>Artiste-qb</td>
</tr>
<tr>
<td>2016</td>
<td>Qubiter</td>
<td>Python</td>
<td>Artiste-qb</td>
</tr>
<tr>
<td>2016</td>
<td>IBM Q Experience</td>
<td>-</td>
<td>IBM</td>
</tr>
<tr>
<td>2017</td>
<td>ProjectQ</td>
<td>Python</td>
<td>ETH Zurich</td>
</tr>
<tr>
<td>2017</td>
<td>Forest (QUIL)</td>
<td>Python</td>
<td>Rigetti</td>
</tr>
<tr>
<td>2017</td>
<td>QISKit</td>
<td>Python</td>
<td>IBM</td>
</tr>
<tr>
<td>2017</td>
<td>Quantum Optics.jl</td>
<td>Julia</td>
<td>Universität Innsbruck</td>
</tr>
<tr>
<td>2017</td>
<td>PsiQuaSP</td>
<td>C++</td>
<td>Gegg M, Richter M</td>
</tr>
<tr>
<td>2018</td>
<td>Strawberry Fields</td>
<td>Python</td>
<td>Xanadu, Canada</td>
</tr>
<tr>
<td>2018</td>
<td>Quantum Dev Kit</td>
<td>Q#</td>
<td>Microsoft</td>
</tr>
<tr>
<td>2018</td>
<td>QCGPU</td>
<td>Rust, OpenCl</td>
<td>Adam Kelly</td>
</tr>
<tr>
<td>2018</td>
<td>NetKet</td>
<td>C++</td>
<td>The Simons Foundation</td>
</tr>
<tr>
<td>2018</td>
<td>OpenFermion</td>
<td>Python</td>
<td>Google, Harvard, UMich, ETH ..</td>
</tr>
</tbody>
</table>

https://github.com/markf94/os_quantumソフトウェア
QuTiP - IMPACT

• RESEARCH • EDUCATION • INDUSTRY

Rényi Entropies from Random Quenches in Atomic Hubbard and Spin Models

A. Elben, B. Vermersch, M. Dalmonte, J. I. Cirac, and P. Zoller
Phys. Rev. Lett. 120, 050406 – Published 2 February 2018

Probing many-body dynamics on a 51-atom quantum simulator
Open source

- **Impact**: Easier to understand and develop an idea with good code and implementation, wider visibility and impact. (e.g., PIQS)

- **Reproducibility**: Faster reproduction of results and application to new problems, data and ideas. (e.g., SciNet, Neural ODE, QGAN)

- **Collaborations and feedback**: Combine efforts and expertise of a wide range of people without barriers. Get feedback, bug reports, suggestions from users.

- **Paper to production**: Stable software implementations can be converted to applications faster. (e.g., Tensor Flow, PyTorch, Scikit-learn)
Talk outline

QuTiP

Open source

GitHub

Travis CI

Sphinx

Read the Docs

Zenodo

conda

• Quantum
• QuTiP intro
• What's new?

• Open source?
• GSoC 2019
Quantum physics
A brief introduction
Superposition

Quantum mechanics describes realities in terms of probability wave functions.

\[ 0 \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}} \]

Why?

Associate wave-like nature to electrons.

Probability wave function to describe states.

Double slit (electrons)

Waves interfere
Quantum mechanics describes realities in terms of probability wave functions.

Wave amplitudes add up
One of the most successful theories out there.

The experiments!

Shut up and calculate
Exponential power of quantum superpositions!

Three qubits can be a superposition of all eight possibilities

\[
|000\rangle + |001\rangle + \ldots + |111\rangle
\]

One shot application of a function to all possible data. Seemingly massive parallelization!

Explore all possibilities simultaneously.

\[ N \sim 2^N \]

But we see only 1s and 0s when we look!

The measurement problem.
Determining an unknown quantum state is tricky. Measurement collapses wave function. **Only a probable answer to the computation.**

- Repeating measurement by making copies is not possible due to no-cloning theorem. Repeat experiment on identically prepared qubits and perform multiple measurements in the end to get the result.

Nature only reveals quantum nature through statistics.

RETHINK
- Noise
- Error correction
- Verification

\[ |000\rangle + |001\rangle + \ldots + |111\rangle \]

\[ \frac{1}{\sqrt{8}} |010\rangle \]
Entanglement

If they measure the same property (basis), they get correlated result. Otherwise, random. Measuring parts, collapses the results.

Correlations between parts $|01\rangle + |10\rangle$

Measurement Basis
- $z$: Up/Down
- $x$: Left/Right

Spooky action at a distance - EPR paradox, Bell inequalities

Spooky! and faster than light communication? (NOT)

Entanglement is a resource: Dense coding, teleportation, quantum key sharing.
What can we do with all that quantum?

- Core idea (Feynman): Simulate quantum physics, atomic and molecular interactions.

- Speed-ups for some problems in Computer Science and Mathematics
  - Integer factorisation (Shor). Break RSA.
  - Grover’s search (Brute force search in unstructured databases).
  - Random sampling of quantum circuits, Boson sampling …

- Optimization and Machine Learning.

- Quantum chemistry, drug design and studying complex biological datasets, protein folding.

- Quantum pattern recognition.

Long way to go. But we can still do interesting things with our computers.
QuTiP
Code
QuTiP speaks quantum

Quantum state vectors

\[ |\psi\rangle = \frac{1}{\sqrt(2)} (|0\rangle + |1\rangle) \]

Operators and Hamiltonians

\[ H = \frac{\sigma_z}{2} \]

Tensors

\[ \sigma_z \otimes \sigma_x \]

Density matrices

\[ \rho \]

>> from qutip import *
>> psi1 = basis(2, 0)
>> psi2 = basis(2, 1)
>> psi = (psi1 + psi2)/1.414

>> H = sigmaz()/2

>> a = destroy(2)

>> tensor(rho1, rho2)
>> tensor(sigmaz(), sigmax())

>> rho = ket2dm(psi)
>> op = vector_to_operator(rho)
The Qobj class

• State and operators are declared as `Qobj`

• Algebra (bosonic)

• Sparse CSR matrices which interact with specialized Cython enhanced code.

```python
>> q = Qobj([1], [0])
Quantum object: dims = [[2], [1]],
shape = (2, 1), type = ket

\[
\begin{pmatrix}
1.0 \\
0.0
\end{pmatrix}
\]

>> d = destroy(2)
Quantum object: dims = [[2], [2]],
shape = (2, 2), type = oper,
isherm = False

\[
\begin{pmatrix}
0.0 & 1.0 \\
0.0 & 0.0
\end{pmatrix}
\]

>> q.dag()
Quantum object: dims = [[1], [2]],
shape = (1, 2), type = bra

\[
\begin{pmatrix}
1.0 & 0.0
\end{pmatrix}
\]
```
<table>
<thead>
<tr>
<th>Operators</th>
<th>Command (# means optional)</th>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charge operator</td>
<td><code>charge(N, M=-N)</code></td>
<td>Diagonal operator with entries from M ... 0 ... N.</td>
</tr>
<tr>
<td>Commutator</td>
<td><code>commutator(A, B, kind)</code></td>
<td>kind = ‘normal’ or ‘anti’ commutator.</td>
</tr>
<tr>
<td>Diagonals operator</td>
<td><code>qdiags(N)</code></td>
<td>Quantum object created from arrays of diagonals at given offsets.</td>
</tr>
<tr>
<td>Higher spin operator</td>
<td><code>jmat(j,#s)</code></td>
<td>j = integer or half-integer representing spin, s = ‘x’, ‘y’, ‘z’, ‘+’, or ‘−’</td>
</tr>
<tr>
<td>Identity</td>
<td><code>qeye(N)</code></td>
<td>N = number of levels in Hilbert space.</td>
</tr>
<tr>
<td>Destruction operator</td>
<td><code>destroy(N)</code></td>
<td>same as above</td>
</tr>
<tr>
<td>Momentum operator</td>
<td><code>momentum(N)</code></td>
<td>same as above</td>
</tr>
<tr>
<td>Number operator</td>
<td><code>num(N)</code></td>
<td>same as above</td>
</tr>
</tbody>
</table>
## Methods on Qobj

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Q.check_herm()</code></td>
<td>Check if quantum object is Hermitian</td>
</tr>
<tr>
<td><code>Q.conj()</code></td>
<td>Conjugate of quantum object.</td>
</tr>
<tr>
<td><code>Q.dag()</code></td>
<td>Returns adjoint (dagger) of object.</td>
</tr>
<tr>
<td><code>Q.diag()</code></td>
<td>Returns the diagonal elements.</td>
</tr>
<tr>
<td><code>Q.eigenenergies()</code></td>
<td>Eigenenergies (values) of operator.</td>
</tr>
<tr>
<td><code>Q.eigenstates()</code></td>
<td>Returns eigenvalues and eigenvectors.</td>
</tr>
<tr>
<td><strong>Q.groundstate()</strong></td>
<td>Eigenval &amp; eigket of Qobj groundstate.</td>
</tr>
<tr>
<td><code>Q.matrix_element(bra,ket)</code></td>
<td>Matrix element $&lt;\text{bra}</td>
</tr>
<tr>
<td><code>Q.norm()</code></td>
<td>Returns L2 norm for states, trace norm for operators.</td>
</tr>
</tbody>
</table>
Superoperators, maps, tensors

- \texttt{spre} and \texttt{spost} \quad X \rho X^\dagger

- Multipartite systems, tensors, two coupled qubits, qubit coupled to an oscillator

\begin{verbatim}
>> tensor(sigmax(), sigmax())
>> tensor(psi1, psi2)
>> tensor(rhol, rho2)
\end{verbatim}

- Composite Hamiltonians, Liouvillians, and super tensors

\begin{verbatim}
>> H_sys = sigmaz()/2
>> H_cav = destroy(5)
>> H_comp = tensor(H_sys, H_cav)
>> super_tensor(L1, L2)
\end{verbatim}

\textbf{Contractions, exotic maps}
Gates as Quantum Objects

• All valid Qobj methods work directly. Eg: Make a super-operator out of Toffoli

\[
\text{Quantum object: } \text{dims = } [[[2, 2, 2], [2, 2, 2]], [[[2, 2, 2], [2, 2, 2]]], \text{shape } = (64, 64), \text{type } = \text{super, isherm } = \text{True}
\]

\[
\begin{pmatrix}
1.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 1.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & \cdots & 0.0 & 0.0 & 0.0 & 0.0 & 1.0
\end{pmatrix}
\]

• Quickly move from the picture of circuits and gates to Hamiltonian evolution and use all of the QuTiP machinery

• Noise, quantum control, stochastic evolution

From circuit to physics
QuTiP

Quantum dynamics
Lindblad master equation solver

- Typical workflow

\[ \dot{\rho} = -i[H, \rho] + \sum_i C_i \rho C_i^\dagger - \frac{1}{2} \{ C_i C_i^\dagger, \rho \} \]

>> H = sigmaz()
>> c_ops = [sigmam()]
>> psi0 = basis(2, 0)
>> times = np.linspace(0.0, 10.0, 20.0)
>> result = mesolve(H, psi0, times, c_ops)

Visualization

# Plot results using Matplotlib
>> plt.plot(result.times, result.expect[0])
>> plt.plot(result.times, result.expect[1])
QuTiP handles time-dependent Hamiltonians with ease

Time-dependent Hamiltonians arise in control problems or driven systems.

Smart built-in solvers such as Floquet formalism: Evolve the wave function "stroboscopically", i.e., only evaluating at time multiples of the driving period.

\[
H = H_0 - f(t)H_1
\]

```python
>> def f(t, args):
    return np.exp(-args[0]*t)

>> H0 = sigmax()/2
>> H1 = sigmaz()/2
>> H = [H0, [H1, f]]

>> mesolve(H, ...)
>> floquet_markov_mesolve
```
Quantum stochastic calculus is the non-commutative analogue of Ito’s calculus, developed to study noise in open quantum systems. (Barichielli, 1990)

Useful tool to model continuous weak measurements, and implement feedback-control methods

For example: Weak continuous Heterodyne and Homodyne measurement techniques (used to extract quadrature information with photon counters)
Other master equation solvers

\[
\dot{\rho} = -\frac{i}{\hbar}[H, \rho] + \sum_{I=1}^{N^2-1} \gamma_i \left( \mathcal{L}_i \rho \mathcal{L}_i^\dagger - \frac{1}{2} \mathcal{L}_i \mathcal{L}_i^\dagger, \rho \right)
\]

```
>> H = sigmaz()
>> c_ops = [sigmam()]
>> psi0 = basis(2, 0)
>> times = np.linspace(0.0, 10.0, 20.0)
>> result = mesolve(H, psi0, times, c_ops)
```

mcsolve  Monte Carlo  HsolverDL  Hierarchy
fmmesolve  Floquet-Markov  ssesolve  Stochastic
rcsolve  Reaction coordinate  brmesolve  Bloch redfield

... and more
Visualisation

Tomography

Bloch Sphere

Wigner

Circuits

Surface code
Topological Circuits
QuTiP: What’s new?
PIQS: simulating qubit ensembles

- PIQS (Permutational Invariant Quantum Solver)
  Nathan Shammah, Shahnawaz Ahmed

  - Noisy driven dissipative open quantum systems
  - Collective effects in qubit ensembles (100)


---

**Ultrastrong Coupling**

![Ultrastrong Coupling Graph](graph1)

**Spin Squeezing**

![Spin Squeezing Graph](graph2)

**Superradiance**

![Superradiance Graph](graph3)
Non-Markovian methods, virtual photons

- A generic Hierarchical Equations of Motion (HEOM) implementation. Neill Lambert, Shahnawaz Ahmed
  - Non-Markovian dynamics: Environment has a memory
  - Ultrastrong coupling regime of light and matter
  - Bound states and virtual photons

A. Fruchtman, N. Lambert, and E.M. Gauger, Scientific Reports, 6, 28204 (2016).
When do perturbative approaches accurately capture the dynamics of complex quantum systems?
Scattering

• Photon scattering in Quantum Optical Systems
  Ben Bartlett, P.h.D student, Stanford University

• Photon scattering
• Multiple waveguides: SPDC, Photon emission


“How do photons scatter into the waveguide when the system is driven with some excitation field?”
Open source and open science
How QuTiP uses open source

Code, Testing

GitHub

Travis CI

Documentation

Sphinx

Read the Docs

Publish

Conda

Zenodo
Github

Online repository

Your fork (copy) on Github

Local copy

• Open a Pull request
• Review by others
• Merge
Testing

"Untested code is broken code"
Documentation

Auto-generate

>> sphinx-quickstart

Edit conf

doc/source/conf.py

# -- Project information
project = 'piqs'

Generate doc

>> make html

piqs.readthedocs.com

Introduction

Permutational Invariant Quantum Solver (PIQS)

PIQS is an open-source Python solver to study the exact Lindbladian dynamics of open quantum systems consisting of identical qubits.
Publishing and distributing code

setup.py  meta.yml

```bash
>> pip install qutip
>> conda -c conda-forge install qutip
>> python setup.py install/develop
```
QuTiP: summing up

The Quantum Toolbox in Python

Authors

Timeline:
Inspired by the Quantum Toolbox in MatLAB.

2011-2012: QuTiP 1.0
Aug 2015: 100 citations
Aug 2016: 200 citations
Jan 2017: QuTiP 4.0
July 2018: QuTiP 4.3

Lead Developers
Alex Pitchford
Aberystwyth University

Contributing Developers
- Neill Lambert (RIKEN)
- Denis Vasilyev (Leibniz)
- Kevin Fischer (Stanford)
- Jonathan Zoller (Ulm University)
- Ben Criger (RWTH Aachen)
- Eric Giguere
- Shahnawaz Ahmed (Chalmers)
- Nathan Shammah (RIKEN)

GitHub: 44 contributors, 4k commits

License: BSD
Style: PEP8 compliant

Libraries used:
- Scipy
- NumPy
- Cython
  - Matplotlib
  - SymPy
  - Jupyter notebooks
  - Online documentation
  - Independent testing

Project Impact
- >600 citations (Google Scholar)
- Downloads 43k (conda forge)

More info at http://qutip.org/

Users
Distribution of 25k website visitors (2016)
We are applying to Google Summer of Code 2019 with NumFocus!

- Lattice models in QuTiP. Ising, Hubbard, XY, Heisenberg model.
  - Clemens Gneiting (Riken, Japan)
  - Eric Giguere (Université de Sherbrooke)

- GPU backend for dynamics with the Hierarchical Eq of Motion
  - Neill Lambert (Riken, Japan)
  - Alex Pitchford (Université de Sherbrooke)

- An overhaul and abstraction of the quantum object class
  - Alex Pitchford (Université de Sherbrooke)
  - Eric Giguère (Université de Sherbrooke, UK)
Thank you

medium.com/quantum-tech

Go to menti.com and use code 54 73 02

sahmed.in
Twitter @quantshah