QuNet: Python library for studying open qubit systems

Documentation and User Guide

Version 1.2

Unnati Akhouri
uja5020@psu.edu
GitHub repository: https://github.com/UnnatiAkhouri/QuNet

Abstract

This documentation and user guide is being created as part of a quantum information bootcamp at the Pennsylvania State University. The four week bootcamp will cover how-to download, run, and analyze qubit networks with varying connectivity. The documentation will be updated routinely on Friday of every week during June 2025.

Contents

1 Introduction to Code

QuNet (previously Qnibble) is a comprehensive Python framework for simulating and analyzing evolution of qubit networks with varying interaction connectivity. This documentation provides a complete guide to understanding, installing, and utilizing the codebase for quantum system simulations.

Throughout the document we will reference papers that facilitate understanding of the various concepts introduced.

1.1 Overview

This package can be used for simulating discrete-in-time evolution of qubit network or circuit with a given or time-varying interaction connectivity. By interaction connectivity which qubits are coupled via interaction terms in the Hamiltonian at a given circuit layer or timestep ℓ. The code was developed to understand the subsystem statistics at the level of one, two and three qubit subsystems within the total system evolving with a Hamiltonian with global symmetry. To that end, the code has particular focus on:

- Qubit Hamiltonians with global conserved quantities
- Qubit networks with varying topology–cyclic chain, fully connected etc.
- Tools to analyze
  - Local, single-qubit dynamical maps
  - Correlation measures like mutual information, concurrence
  - Principal component analysis of local quantities
  - Network measures on mutual information networks like clustering and disparity
  - Thermodynamic quantities like relative entropy and extractable work.
- Visualization of quantum trajectories

1.2 Key Features

- Efficient simulation of qubit networks with modular topology
- Support for custom Hamiltonian definitions
- Built-in visualization tools
- Extensible toolkit for analyzing local subsystem statistics
- Ability to run on cluster and HPC
- Open source

2 Motivation

Can a closed quantum system, evolving under global conservation laws, give rise to subsystems that exhibit persistent out-of-equilibrium dynamics? If so, what is the role of the conserved quantity and do all Hamiltonians within a class of Hamiltonians obeying a symmetry exhibit this or does connectivity have a role to play?

To investigate these questions, we developed a code to study systems can evolve to display different classes of subsystem dynamics over extended periods, a behavior clearly distinguishable

from that of approximately thermalizing networks of comparable size. Such quantum systems are fundamental for understanding the emergence of complex and out of equilibrium behavior, with potential connections to biological and condensed matter systems.

In a recent work, we used this code to introduce a novel class of constrained dynamics wherein the quantum system optimizes thermodynamic or quantum information-theoretic measures to explore non-equilibrium state space. The systems studied retain local memory of their initial conditions.

The code allows us to characterize the subsystems by bringing in tools from quantum information—namely phase-covariant dynamical maps—and the theory of open quantum systems. In this way, the subsystems can be studied as an ensemble of open systems, specifically a collection of qubits evolving with phase-covariant dynamics.

Although constraints imposed by the conservation law and the global unitary dynamics of the network bound the distribution of single-qubit dynamics within the ensemble, distinct steady states remain distinguishable by multiple metrics, offering insights into the fundamental behavior of quantum systems far from equilibrium.

By studying the local maps as well as the correlation structure on the qubit network we can draw conclusions about the role correlations play in the long-term local evolution.

Our hope is that this code provides a computational framework for investigating emergent phenomena in quantum statistical mechanics and allows us to examine how constraints and conservation laws can give rise to persistent non-equilibrium states and novel dynamics.

2.1 Scientific Background

The study of qubit system evolution is fundamental to understanding quantum computing, quantum information processing, and quantum many-body physics. Existing simulation tools, often, either are oriented to answer questions ab

2.2 Research Applications

This framework enables researchers to:

- Investigate quantum coherence in multi-qubit systems
- Study the effects of environmental decoherence
- Optimize quantum control protocols
- Validate theoretical predictions with numerical simulations

3 Setup

3.1 Prerequisites

Before installing QuNet, ensure you have the following requirements:

Installed Python Packages

3.2 Installation

We recommend that you use Github desktop for interfacing with the code, especially if you intend to make changes/modify the code. Additionally having a python editor platform like PyCharm will be beneficial if you are to read and edit the code.

Package	Version
collections-extended	2.0.2
fqdn	1.5.1
isoduration	20.11.0
jsonpointer	3.0.0
jupyter	1.1.1
mpl-toolkits.clifford	0.0.2
networkx	3.4.2
pip	23.2.1
plotly	6.1.2
scikit-learn	1.6.1
seaborn	0.13.2
statsmodels	0.14.4
tinycss2	1.4.0
tqdm	4.67.1
uri-template	1.3.0
webcolors	24.11.1
wheel	0.41.2

Table 1: Installed Python Packages

3.2.1 From Source

```
git clone https://github.com/UnnatiAkhouri/QuNet.git
cd qunet
pip install -e .
```

3.3 Verification

To verify your installation:

```
import qunet
print(qunet.__version__)
qunet.test()  # Run basic tests
```

4 Creating Virtual Environment

Upon opening the github repository in Pycharm or whichever platform you use, navigate the path to create a virtual environment if not already prompted by the application. In Pycharm, you can follow the following steps to ensure you create a proper virtual environment. Creating a virtual environment ensures that the packages and the version of the packages, including python itself, needed for this library do not interfere with the python and packages on your base. Here are the steps:

1. Go to PyCharm → Settings
2. In the pop-up widow click the project denoted by the project name
3. Click on python interpreter → Add interpreter
4. Click Add local interpreter
5. In the dialog that opens:

- Select Virtualenv Environment from the left panel
- Choose New environment.
- Select a location for your virtual environment (usually in your project folder)
- Choose the base Python interpreter i.e. verison of python
- Click Ok/apply

4.1 Known Issues

- If Jupyter plugin has not been downloaded on pycharm, you will not be able to run the notebooks in pycharm
- If packages have not been downloaded properly, many code functions fail to run. To check which libraries you have, in the terminal type **pip list –not-required** This will give you a list of all the packages you have in the venv.
- matplotlib does not appear as a separate package in the list because it is a dependency of seaborn and plotly.

5 GitHub Files

5.1 Repository Structure

The QuNet repository is organized as follows:

```
qunet/
           README.md
           LICENSE
           Scripts
           Singularity
           src
           data
           Tests
                 test_comprehensive.py
                 test_density_matrix.py
                 test_ket.py
                 test_measurements.py
           notebooks/
               many analysis jupyter notebooks
               .ipynb
               images get stored here
```

5.2 Key Files Description

5.2.1 Core Module Files

-
-
-

5.2.2 Utility Files

-
-

5.2.3 Documentation and Examples

- `notebooks/`: Jupyter notebooks with tutorials and examples
- `docs/`:
- `examples/`: Python script examples

5.3 Contributing Guidelines

To contribute to QuNet:

1. Fork the repository
2. Create a feature branch: `git checkout -b feature-name`
3. Make your changes and add tests
4. Run the test suite: `pytest`
5. Submit a pull request

6 Analysis

6.1 Theory

6.1.1 Mathematical Framework

6.1.2 Numerical Methods

6.1.3 Multi-Qubit Systems

For N-qubit systems, the Hilbert space dimension scales as 2^N. QuNet handles this through:

- Efficient sparse matrix representations
- Tensor product operations
- Memory-optimized state vector storage

6.2 Notebooks

The QuNet package includes comprehensive Jupyter notebooks for learning and experimentation:

The analysis notebooks assume that the data is stored in the hdf5 file format. This will be the case if you choose to use this code to develop your data via local-on-machine simulation or on-cluster simulation. However, if you want to use the code to analyze external data, not in the hdf5 format, you will have to modify the functions discussed in this section.

For more information on the hdf5 file format read this. `https://www.earthdata.nasa.gov/about/esdis/esco/standards-practices/hdf5`

6.2.1 Tutorial Notebooks

Basic Tutorial (`tutorial_basic.ipynb`) This notebook covers:

- Setting up single and multi-qubit systems
- Defining custom Hamiltonians
- Basic time evolution

- Visualization of quantum trajectories
- Computing expectation values

Example workflow:

```
import qunet as qn
import numpy as np

# Create a two-qubit system
system = qn.QubitSystem(n_qubits=2)

# Define initial state |01
psi0 = qn.basis_state([0, 1])

# Create Hamiltonian (e.g., Ising model)
H = qn.Hamiltonian.ising(coupling=0.1, field=0.05)

# Time evolution
times = np.linspace(0, 10, 100)
evolution = qn.evolve(psi0, H, times)

# Visualize results
qn.plot_populations(evolution, times)
```

Advanced Tutorial (`tutorial_advanced.ipynb`) Advanced topics include:

- Time-dependent Hamiltonians
- Tz shift parameter
- Concurrence
- MI networks
- MMI

6.2.2 Example Notebooks

Physics Applications

-
-

Benchmarking and Validation

-

7 Conclusion

QuNet provides a robust, well-documented framework for quantum system simulation. The combination of efficient numerical methods, comprehensive documentation, and interactive tutorials makes it suitable for both educational use and research applications.

For the latest updates, bug reports, and feature requests, please visit the GitHub repository at `https://github.com/UnnatiAkhouri/QuNet`.

References