A Quantum Toolchain

Michael H. Goerz   http://michaelgoerz.net   @goerz   github.com/goerz   github.com/mabuchilab

current: U.S. Army Research Lab, Computational and Information Science Directorate, Adelphi, MD *previous:* Ginzton Lab, Stanford University, Stanford, CA

QNET   Computer Algebra for Quantum Mechanics

QNET has strong connection to
- SymPy (Python's general computer algebra library)
- QuTiP (Quantum Toolbox in Python)

Connections to SymPy:
- use SymPy for scalar algebra
- similar tree-structure of expressions
- printing system mirrors SymPy

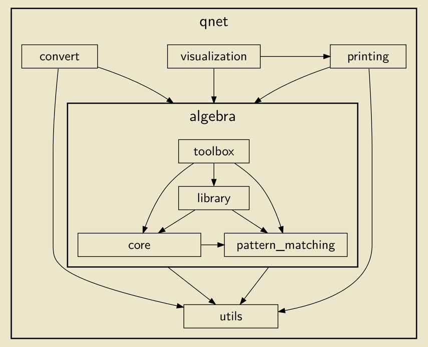
SymPy has some (incomplete) support for quantum algebra in 'sympy.physics.quantum'

Connections to QuTiP:
- use QuTiP for simulation and converting to numerical representations for HPC backends
- similar semantics (e.g. QNET 'Create' → QuTiP 'create')
- some method names mirror QuTiP

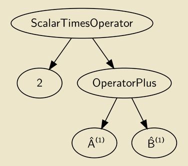
Differences to SymPy:
- semantics and notation of quantum mechanics
- no automatic simplifications on: "smart" instantiation via 'create' method
- focus on exact expression tree manipulation
- improved pattern matching
- easier to debug and extend (no __new__)

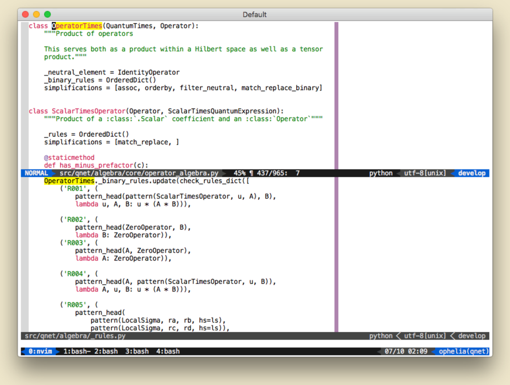
Differences to QuTiP:
- expressions are analytic: unknown/infinite bases allowed
- quantum objects have an explicit associated Hilbert space: tensor products are implicit

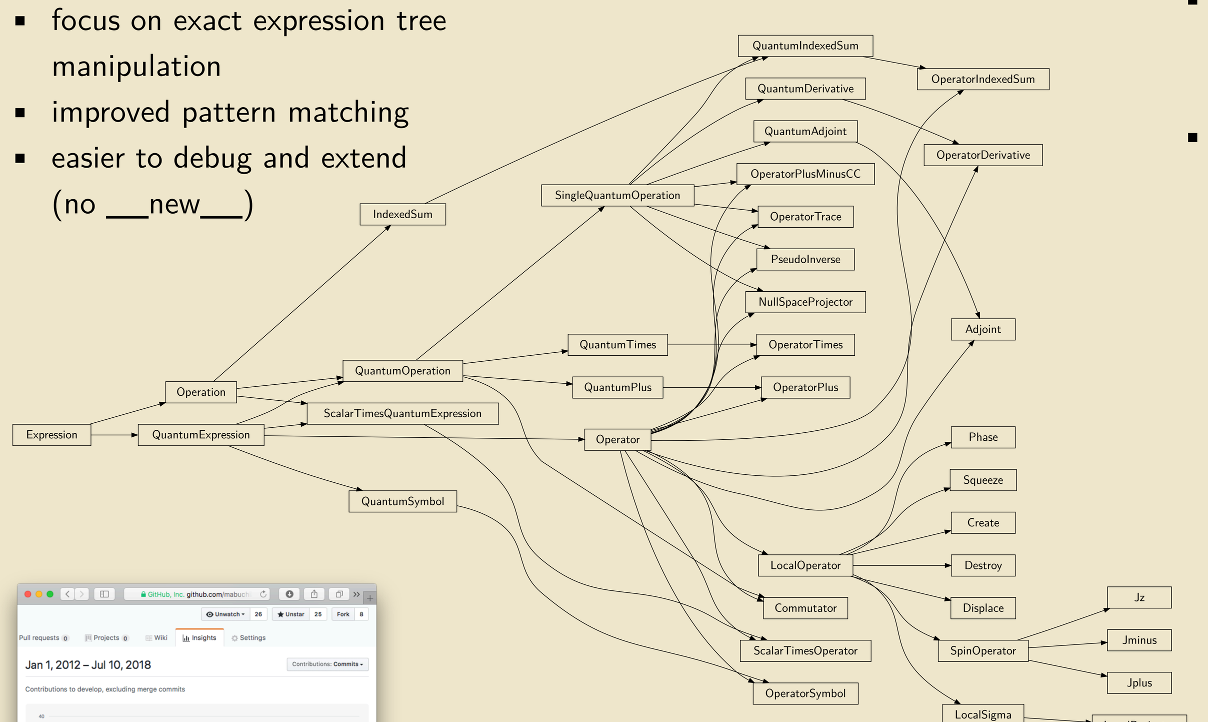
Library package-structure:
- algebra.core: quantum algebra class hierarchy
- algebra.library: domain-specific extensions
- algebra.pattern_matching: pattern-based manipulation of expression trees
- algebra.toolbox: analytical algorithms (e.g. perturbation theory)
- convert: conversion to QuTiP, SymPy matrices
- visualization: circuit diagrams
- printing: customizable printing (unicode / LaTeX)



Core algebra:
- scalar algebra: wrapper around numeric values, SymPy; but also "scalar expressions", e. g. $\langle\Psi_1|\Psi_2\rangle$
- operator-algebra: full C*-algebra
- state-algebra: Hilbert-space algebra of quantum states
- super-operator-algebra: Liouville space algebra for open quantum systems
- circuit-algebra: combine small quantum systems in a network, according to "SLH" algebra.





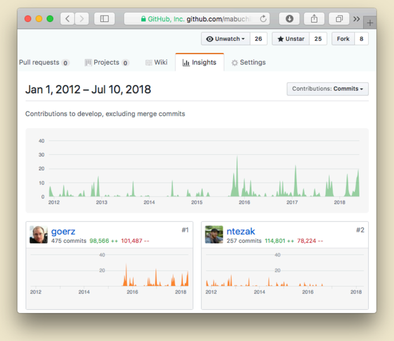


concatenation: $(\mathbf{S}_1, \mathbf{L}_1, H_1) \boxplus (\mathbf{S}_2, \mathbf{L}_2, H_2) = \left(\begin{pmatrix} \mathbf{S}_1 & 0 \\ 0 & \mathbf{S}_2 \end{pmatrix}, \begin{pmatrix} \mathbf{L}_1 \\ \mathbf{L}_1 \end{pmatrix}, H_1 + H_2 \right)$

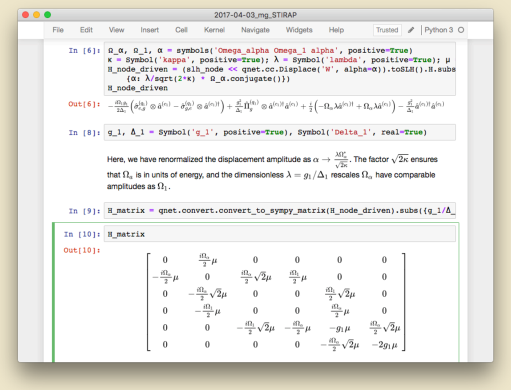
series product: $(\mathbf{S}_2, \mathbf{L}_2, H_2) \lhd (\mathbf{S}_1, \mathbf{L}_1, H_1) = \left(\mathbf{S}_2\mathbf{S}_1, \mathbf{L}_2 + \mathbf{S}_2\mathbf{L}_1, H_1 + H_2 + \operatorname{Im}\left\{ \mathbf{L}_2^\dagger \mathbf{S}_2 \mathbf{L}_1 \right\} \right)$

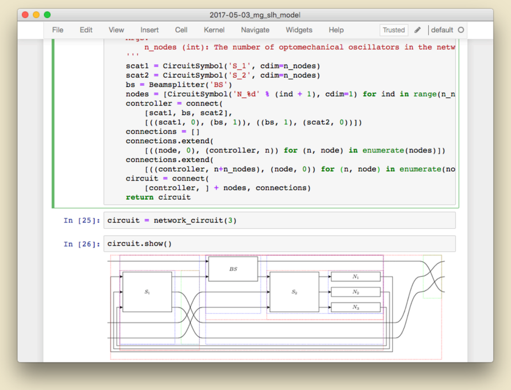
feedback: $[\,(\mathbf{S}, \mathbf{L}, H)\,]_{k\to l} = \left(\tilde{\mathbf{S}}, \tilde{\mathbf{L}}, \tilde{H} \right)$,

QNET is the only software package implementing the SLH circuit algebra!



active development: release 2.0 out soon!
contributions welcome!





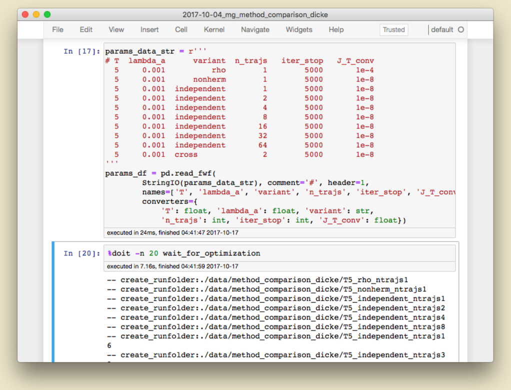
clusterjob   Wrapper around HPC job scripts and interface to local or remote schedulers

Supported job schedulers (backends): SLURM, PBS Pro, PBS/TORQUE, LSF — easy to extend!

Features:
- Use scheduler-agnostic job scripts (shell script)
- Resource header comments automatically added depending on backend
- Allow for MPI, OpenMP, and hybrid parallelization
- Separate job description from backend/resource configuration via config files
- Submit job script to local or remote scheduler via ssh
- Prologue and epilogue hooks for setup / data retrieval
- Asynchronous management of jobs. Compatible with mulitprocessing and ipyparallel
- Caching: connections to schedulers can be resumed

DoIt Automation Tool

pydoit workflow in jupyter notebook:
- pandas-table of parameters to vary
- job: for any new/changed row in table
  - generate an analytic model
  - convert and write input data for QDYN simulation/optimization
- job: for any input data on disk without matching output data: generate and submit clusterjob
- job: wait for any running simulation jobs to complete on cluster; fetch output data when finished



Notebook can be stopped/rerun at any time: final state is "idempotent"

The Toolchain Approach

- every aspect of the research project is represented in software
- interoperability between packages across platforms and languages
- driven by Jupyter notebooks or scripts

goal: reproducibility, correctness, iterative analysis, "intractable" models

Applications

- quantum computing: design "quantum gates"
- quantum sensors: distributed quantum systems
- teaching: a complete theoretician's toolbox for quantum mechanics

gEDA

QNET python

Design and analysis of photonic circuit models
- QHDL model
- SLH formalism
- symbolic quantum algebra
- circuit component library
- visualization
- drive numerical backends

https://github.com/mabuchilab/qnet

jupyter
sympy
matplotlib

QMANIFOLD julia
- Symbolic tools for transforming open quantum system models to a moving basis parametrized by semi-classical coordinates.
- Numerical tools for performing quantum diffusion simulations.

QuTiP

QSD C++

Moving basis solver for quantum trajectories:
- diffusive unravelings of the ME
- accumulate trajectories to obtain expectation values

https://github.com/mabuchilab/qsd-mpi

QDYN QDYN

Fortran library:
- efficient simulation of quantum dynamics
- gradient-based optimization methods

https://qdyn-library.net

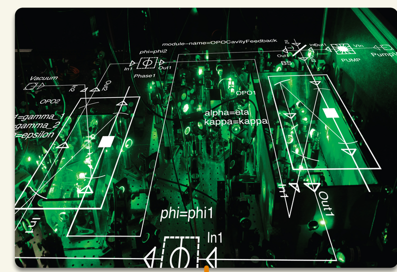
clusterjob python

Create and manage jobs on a HPC cluster

https://github.com/goerz/clusterjob

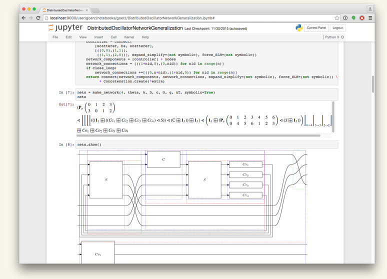
Description of Experiment

- QHDL: describe physical circuit in terms of components and connections
- SLH formalism: description for network can be derived algebraically from components



Analysis

- Model reduction
  - adiabatic elimination
  - coherent manifold approach
- Steady-states, semi-classical fixpoints



HPC Simulation & Optimization

- Quantum Dynamics
  - Simulate:
    - "Schrödinger" equation
    - "Liouville" equation
    - stochastic differential equations
- Optimization: controls that drive system in some desired way

"Design Loop":
- accurate modeling of experimental setups
- theoretical analysis using computer algebra
- numerical simulation of system behavior
- optimization for automated design
- closing the loop: simulation results inform new experimental setups