Quantum algorithm for open systems using noise

HQS QUANTUM SIMULATIONS

A quantum algorithm for solving open system dynamics on quantum computers using noise

Juha Leppäkangas, Nicolas Vogt, Keith R. Fratus, Kirsten Bark, Jesse A. Vaitkus, Pascal Stadler, Jan-Michael Reiner, Sebastian Zanker, and Michael Marthaler
HQS Quantum Simulations GmbH, Rintheimer Str. 23, 76131 Karlsruhe, Germany


Juha Leppäkangas


Nicolas Vogt


Keith R. Fratus


Kirsten Bark


Jesse A. Vaitkus


Pascal Stadler


Jan-Michael Reiner


Sebastian Zanker


Michael Marthaler

European Commission
Horizon 2020
European Union funding
for Research & Innovation

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

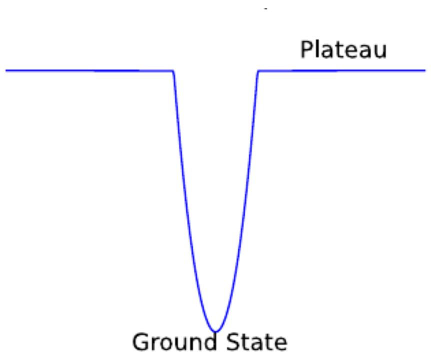
Gefördert durch:
Bundesministerium
für Wirtschaft
und Klimaschutz
aufgrund eines Beschlusses
des Deutschen Bundestages

AVaQus
QSOLID
Q-EXA
PlanQK

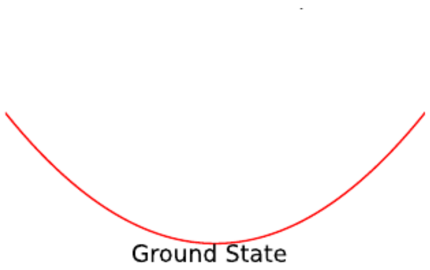
Why non-unitary evolutions matter

- Barren plateaus are a problem in finding the true ground state or global minimum with a Quantum computer
- But nature can find stable local minima when cooling a system
- In nature a system loses energy to a bath with a non-unitary evolution

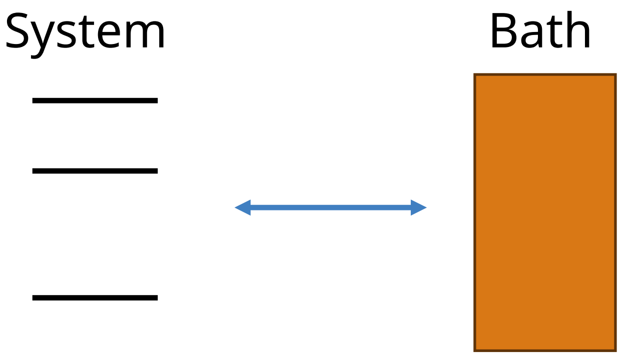
Unitary



Non-Unitary



Local Minima in Quantum Systems
C.-F. Chen, H.-Y. Huang, J. Preskill, L. Zhou
Proceedings of the 56th Annual ACM Symposium on Theory of Computing



Open-System Basics

- Open Systems are given by a quantum system coupled to a much larger quantum bath
- Described by
  1) the Bloch-Redfield equation with a System, a spectral function and coupling operator

$$\mathrm{H}_{\mathrm{system}} \quad S_{\hat{x}_1,\hat{x}_2}(\omega) \quad \hat{x}_1, \hat{x}_2, \ldots$$

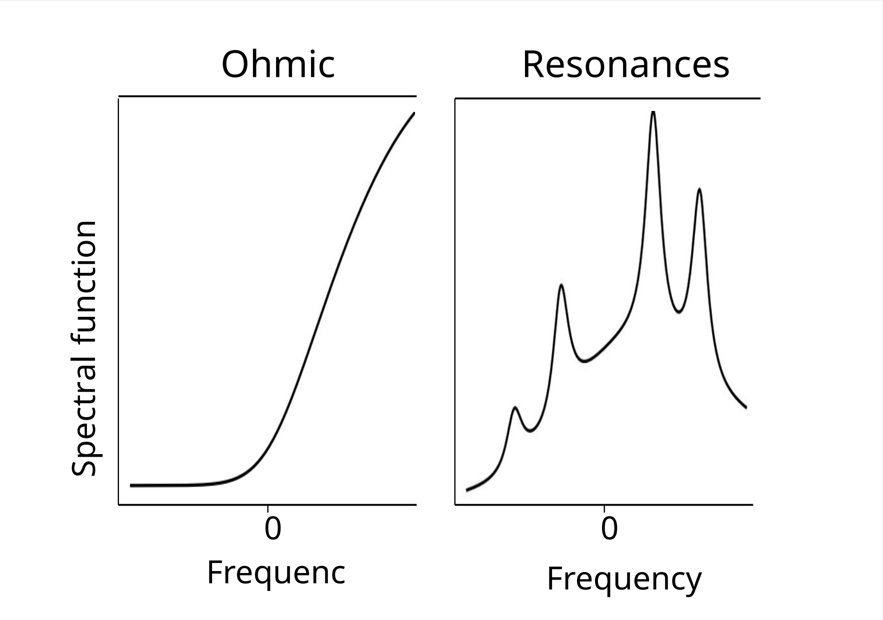
  2) The Lindblad equation

$$\dot{\rho} = i\,[\rho, H] + \sum_{k,l} \Gamma_{k,l} L_k \rho L_l^\dagger - \frac{1}{2}\left\{L_l^\dagger L_k, \rho\right\}$$

$$H_0 = H_S + H_C + H_B$$

$$= \underbrace{\frac{\epsilon}{2}\sigma_z}_{H_S} + \underbrace{\sum_k v_k \sigma_x \left(b_k^\dagger + b_k\right)}_{H_C} + \underbrace{\sum_k \omega_k b_k^\dagger b_k}_{H_B}$$

$$S_{\mathrm{o}}(\omega) = \frac{4\pi\hbar^2\alpha\omega}{1 - \exp(-\frac{\hbar\omega}{k_B T})}$$



Tooling to deal with open systems

- Representation of coupled subsystems

$$H = H_{\mathrm{S}} + H_{\mathrm{C}} + H_{\mathrm{B}}$$

- Representation and manipulation of spectral functions (equivalent to full BR)

$$S_{\hat{x}_0,\hat{x}_1}(\omega)$$

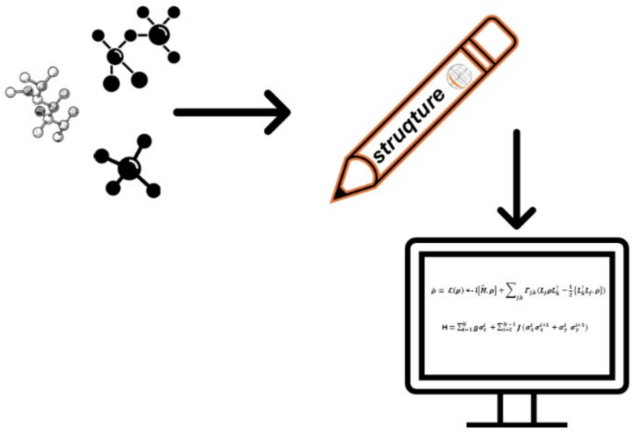
- Representation of Lindblad Systems

$$\dot{\rho} = i\left[\rho, H\right] + \sum_{k,l} \Gamma_{k,l} L_k \rho L_l^{\dagger} - \frac{1}{2}\left\{L_l^{\dagger} L_k, \rho\right\}$$

Struqture

Struqture is a Rust (struqture) and Python (struqture-py) library by HQS Quantum Simulations to represent quantum mechanical operators, Hamiltonians and open quantum systems. The library supports building spin objects, fermionic objects, bosonic objects and mixed system objects that contain arbitrary many spin, fermionic and bosonic subsystems.

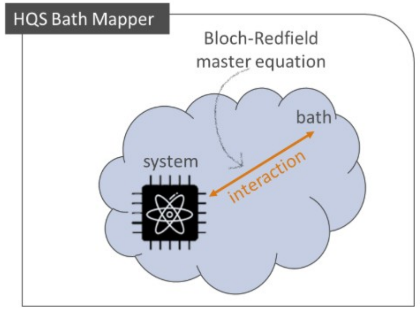
Struqture has been developed to create and exchange definitions of operators, Hamiltonians and open systems. A special focus is the use as input to quantum computing simulation software.



Bath Mapper User Guide

Background physics

The Bath Mapper is a module that can be used in conjunction with other Quantum Libraries by HQS Quantum Simulations GmbH. It supports the user in creating quantum mechanical objects which utilize the Bloch-Redfield master equation. This equation is a mathematical model, here used in quantum computing, to describe the dynamics of open quantum systems interacting with their environment (or "bath"). The Bath Mapper also provides functionality to allow further processing of the created objects.



Tooling to deal with open systems

- Representation of coupled subsystems

$$H = H_{\mathrm{S}} + H_{\mathrm{C}} + H_{\mathrm{B}}$$

- Representation and manipulation of spectral functions (equivalent to full BR)

$$S_{\hat{x}_0,\hat{x}_1}(\omega)$$

- Representation of Linbldad Systems

$$\dot{\rho} = i\left[\rho, H\right] + \sum_{k,l} \Gamma_{k,l} L_k \rho L_l^{\dagger} - \frac{1}{2}\left\{L_l^{\dagger} L_k, \rho\right\}$$

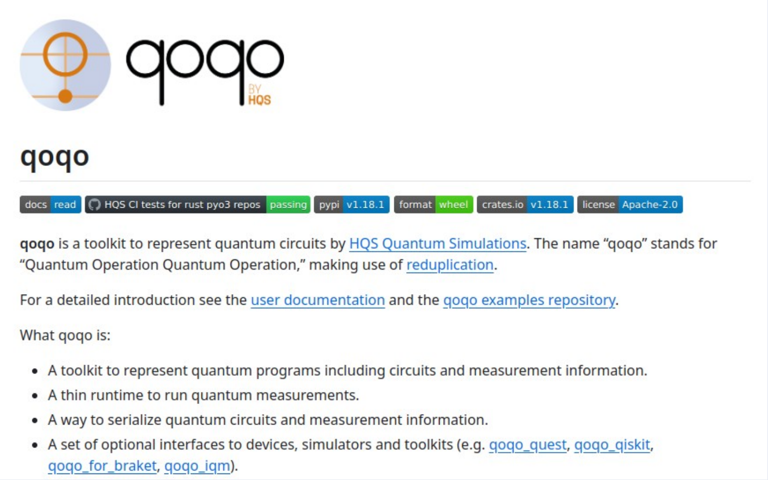
```
from struqture_py.mixed_systems import (
    MixedHamiltonian,
    HermitianMixedProduct
)
from struqture_py.spins import PauliProduct
from struqture_py.bosons import BosonProduct
spin_boson_hamiltonian = MixedHamiltonian(1,1,0)

index = HermitianMixedProduct([PauliProduct().z(0)],
                              [BosonProduct([0], [0])],
                              [])
spin_boson_hamiltonian.set(index, 0.5)
```

```
from bath_mapper import SpinBRNoiseOperator
import numpy as np

frequencies = np.arange(0, 10, 0.1)
spectrum_array  = np.arange(0, 10, 0.1)
# Create spin-specturm with coupling from input.
spectrum = SpinBRNoiseOperator(frequencies)
spectrum.set(("0X", "0X"), spectrum_array)
```

```
from struqture_py.spins import (
    QubitLindbladNoiseOperator,
)

op = QubitLindbladNoiseOperator()
op.set(("0X", "0X"), 0.1)
```

Non-unitary operations in quantum circuits

- Need to be treated on same level as unitary operations
- Apply superoperator to density matrix instead of unitary matrix to state
- Typical noise is a subset



qoqo

docs read | HQS CI tests for rust pyo3 repos passing | pypi v1.18.1 | format wheel | crates.io v1.18.1 | license Apache-2.0

qoqo is a toolkit to represent quantum circuits by HQS Quantum Simulations. The name "qoqo" stands for "Quantum Operation Quantum Operation," making use of reduplication.

For a detailed introduction see the user documentation and the qoqo examples repository.

What qoqo is:

- A toolkit to represent quantum programs including circuits and measurement information.
- A thin runtime to run quantum measurements.
- A way to serialize quantum circuits and measurement information.
- A set of optional interfaces to devices, simulators and toolkits (e.g. qoqo_quest, qoqo_qiskit, qoqo_for_braket, qoqo_iqm).

```python
from qoqo import Circuit
from qoqo import operations as ops

non_unitary = ops.PragmaDamping(
    qubit=0,
    gate_time = 0.1,
    rate = 1e-4
)

circuit = Circuit()
circuit += non_unitary

print(non_unitary.superoperator())
```

✓ 0.0s

```
[[1.00000e+00 0.00000e+00 0.00000e+00 9.99995e-06]
 [0.00000e+00 9.99995e-01 0.00000e+00 0.00000e+00]
 [0.00000e+00 0.00000e+00 9.99995e-01 0.00000e+00]
 [0.00000e+00 0.00000e+00 0.00000e+00 9.99990e-01]]
```

Non-unitary evolutions on quantum computers

An implementation of a non-unitary (system-bath) evolution on a quantum computers needs non-unitary gates. Two main approaches proposed:

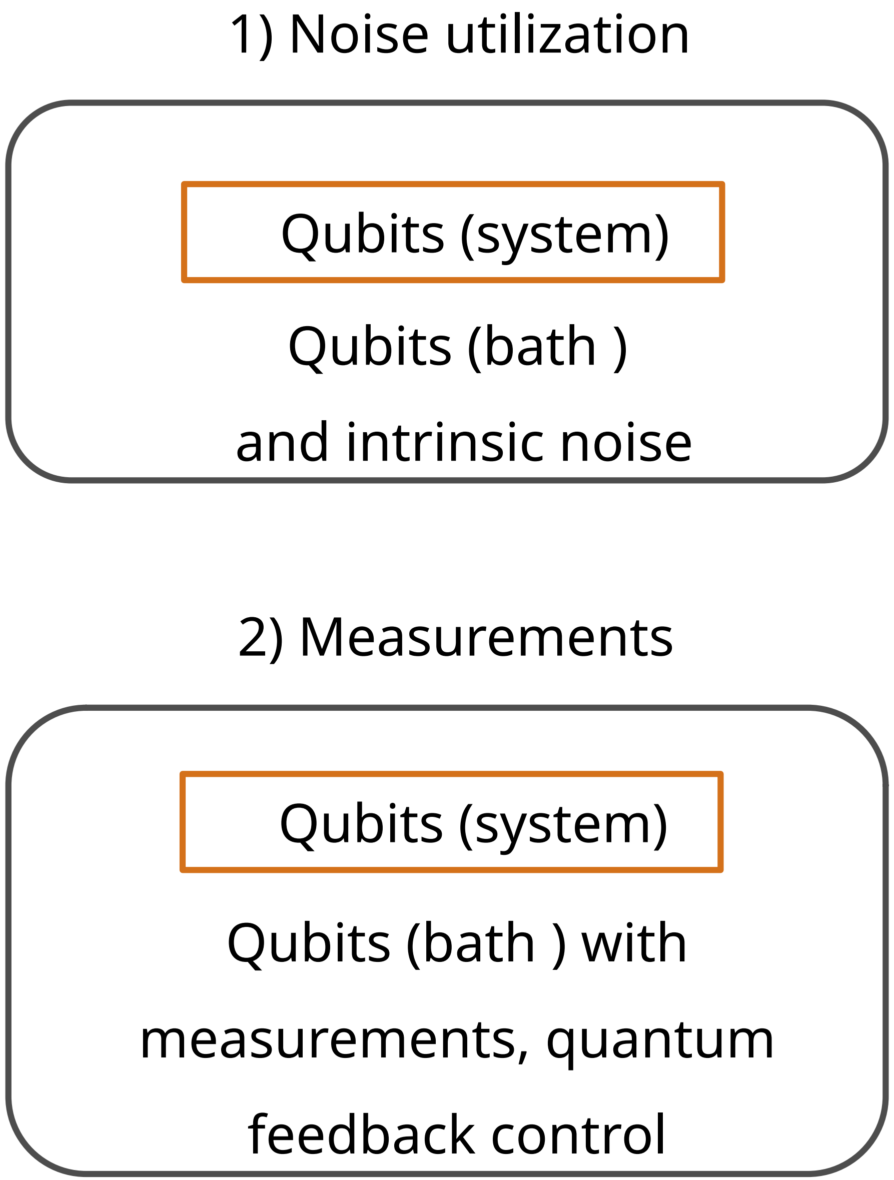
1) Noise utilization:

   Mapping non-unitary gates to intrinsic noise

2) Measurements:

   Coupling to external qubits, performing measurements, quantum feedback control

Stable Quantum-Correlated Many Body States via Engineered Dissipation, https://arxiv.org/abs/2304.13878



Open quantum system models

$$H = H_{\mathrm{S}} + H_{\mathrm{C}} + H_{\mathrm{B}}$$

$$= \underbrace{\sum_i \left(\frac{\epsilon_i}{2} \sigma_z^i + \frac{\Delta_i}{2} \sigma_x^i \right) + \sum_{ij} J_{ij} \sigma_z^i \sigma_z^j}_{H_S} + \underbrace{\sum_{ik} \vec{v}_{ik} \vec{\sigma}_i \left(b_k^\dagger + b_k \right)}_{H_C} + \underbrace{\sum_k \omega_k b_k^\dagger b_k}_{H_B}$$

The bath is described by the

spectrum:

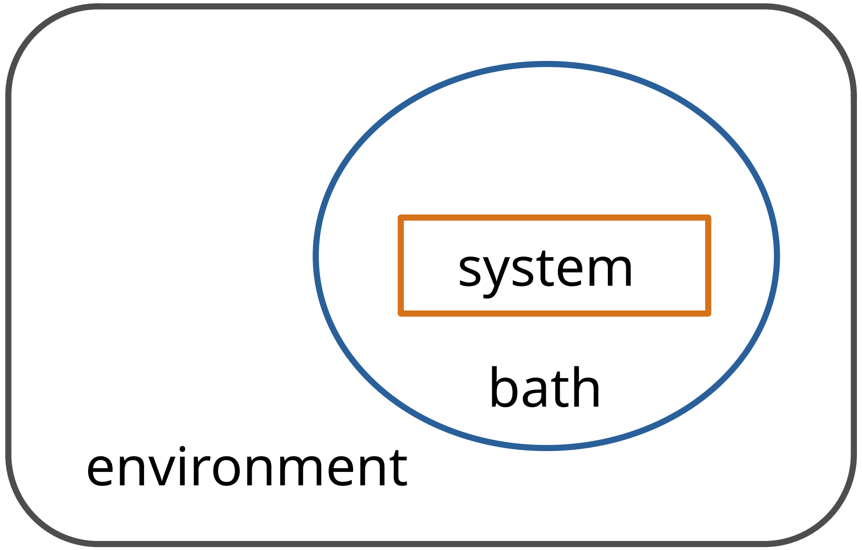
$$S(\omega) = \frac{\sum_{k=1}^{\infty} v_k^2 \delta(\omega - \omega_k)}{1 - \exp\left(-\frac{\omega}{k_{\mathrm{B}} T} \right)} \mathrm{sign}(\omega)$$

couplings

$$= \frac{1}{\pi} \sum_{i=1}^{n} v_i^2 \frac{\kappa_i / 2}{(\kappa_i/2)^2 + (\omega - \omega_i)^2}$$

broadenings

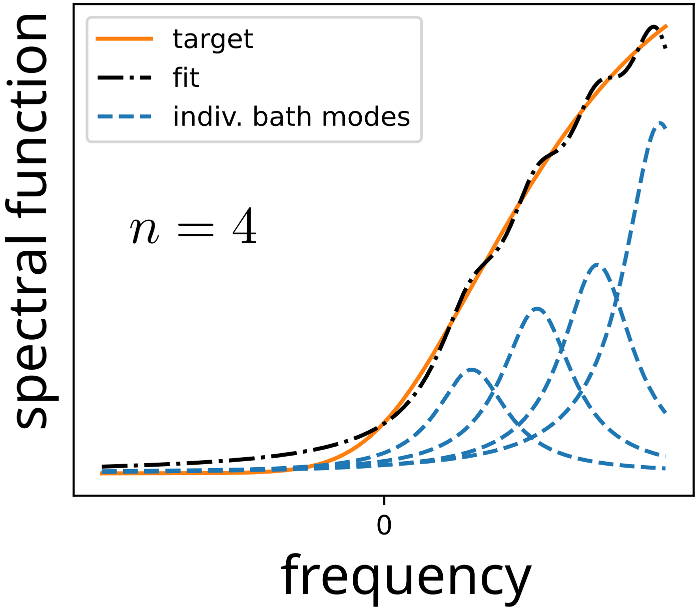
mode frequencies



Coarse graining

- Spectral function fitted by broad modes (Lorentzians)
- Broad modes identified as *auxiliary* boson modes
- In particular, broadening $\kappa/2$ can be mapped to auxiliary-mode damping with rate κ

Coarse graining by four modes



couplings

$$S(\omega) = \frac{1}{\pi} \sum_{i=1}^{n} v_i^2 \frac{\kappa_i/2}{(\kappa_i/2)^2 + (\omega - \omega_i)^2}$$

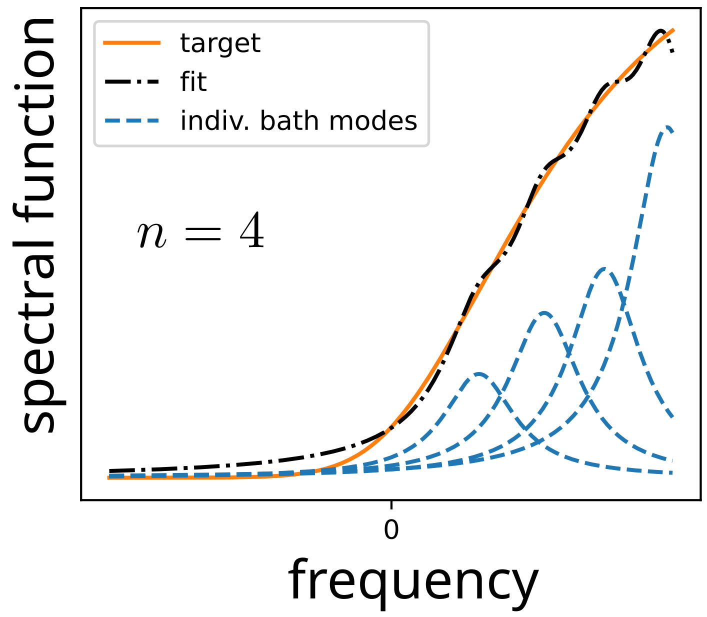
broadenings

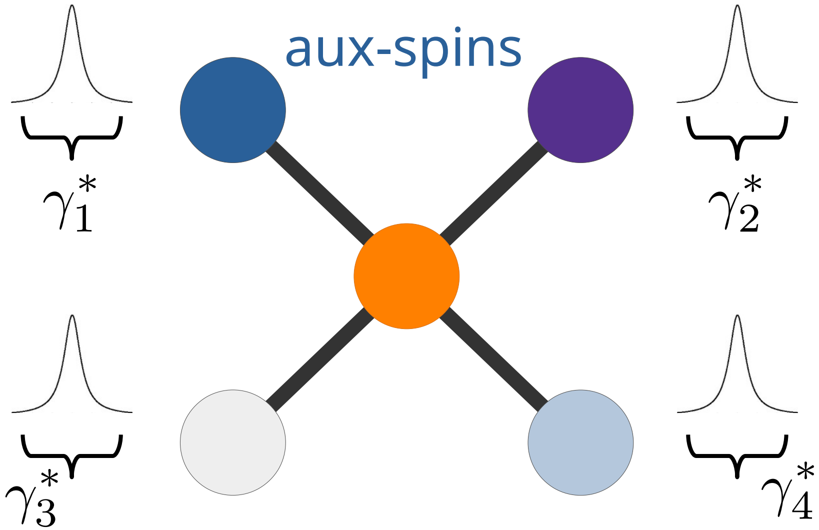
mode frequencies

Possible approach

- One-to-one correspondence between the boson modes and auxiliary spins
- Bath gaussianity improved by letting the broad spin-modes overlap
- Applied if the device has widely distributed decoherence rates of bath qubits

Coarse graining using four qubits





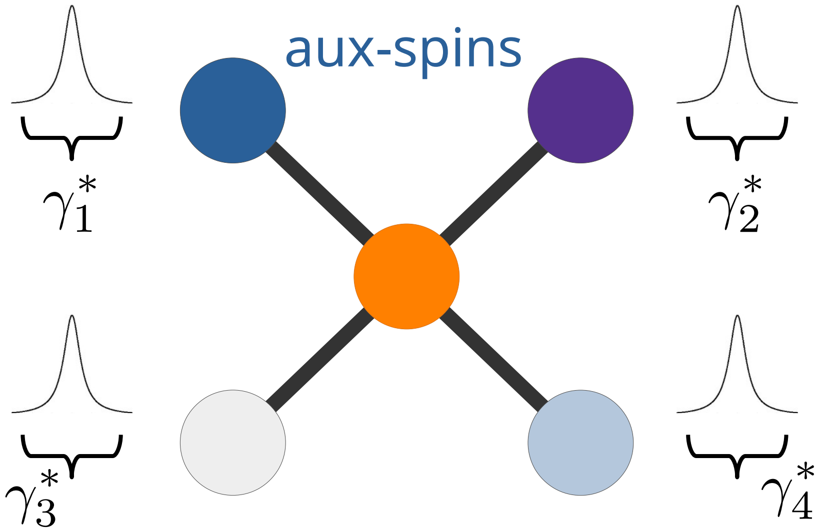
Coherent time-evolution implemented by unitary gates

Time-evolution operator

$$U = \left[\exp\left(-\mathrm{i}\hat{H}\tau\right)\right]^m$$

Total simulation time

$$T = \tau m$$



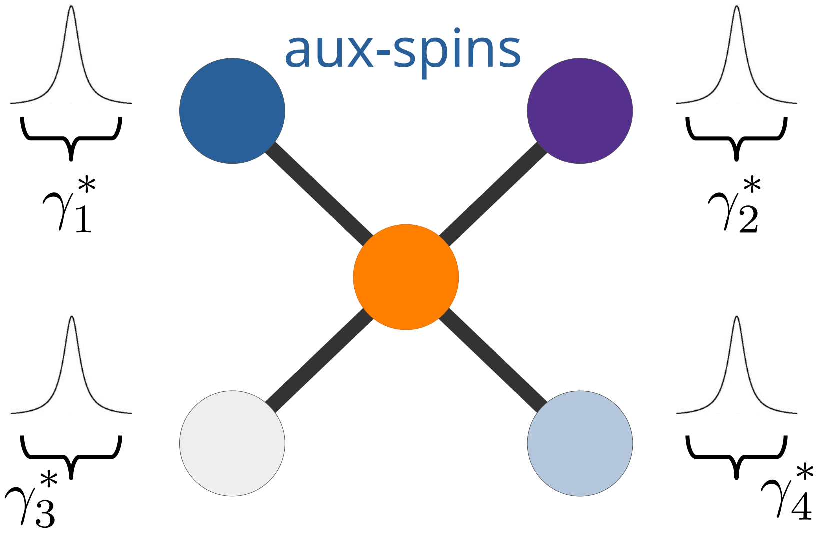
Spin-spin Hamiltonian (approach 1)

$$\hat{H} = \frac{\Delta}{2}\sigma_z + \sigma_x \underbrace{\sum_{i=1}^{n} \frac{v_i}{\sqrt{N_i}} \sum_{j=1}^{N_i} \sigma_x^{ij}}_{v_i\left(b_i + b_i^\dagger\right)} + \underbrace{\sum_{i=1}^{n} \omega_i \sum_{j=1}^{N_i} \frac{\sigma_z^{ij}}{2}}_{\omega_i b_i^\dagger b_i}$$

Non-unitary time-evolution by intrinsic noise

$$\dot{\rho} = \mathrm{i}[\rho, H]$$

$$+ \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{N_i}\gamma_i\left(\sigma_-^{ij}\rho\sigma_+^{ij} - \frac{1}{2}\sigma_+^{ij}\sigma_-^{ij}\rho - \frac{1}{2}\rho\sigma_+^{ij}\sigma_-^{ij}\right)}_{\mathcal{L}_1\rho}$$

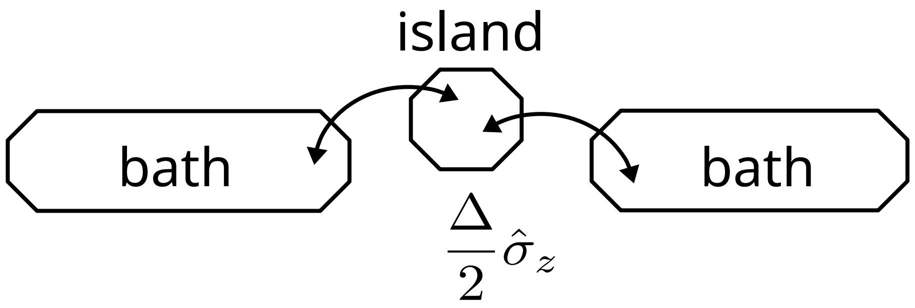


$$+ \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{N_i}\Gamma_i\left(\sigma_z^{ij}\rho\sigma_z^{ij} - \rho\right)}_{\mathcal{L}_2\rho}$$

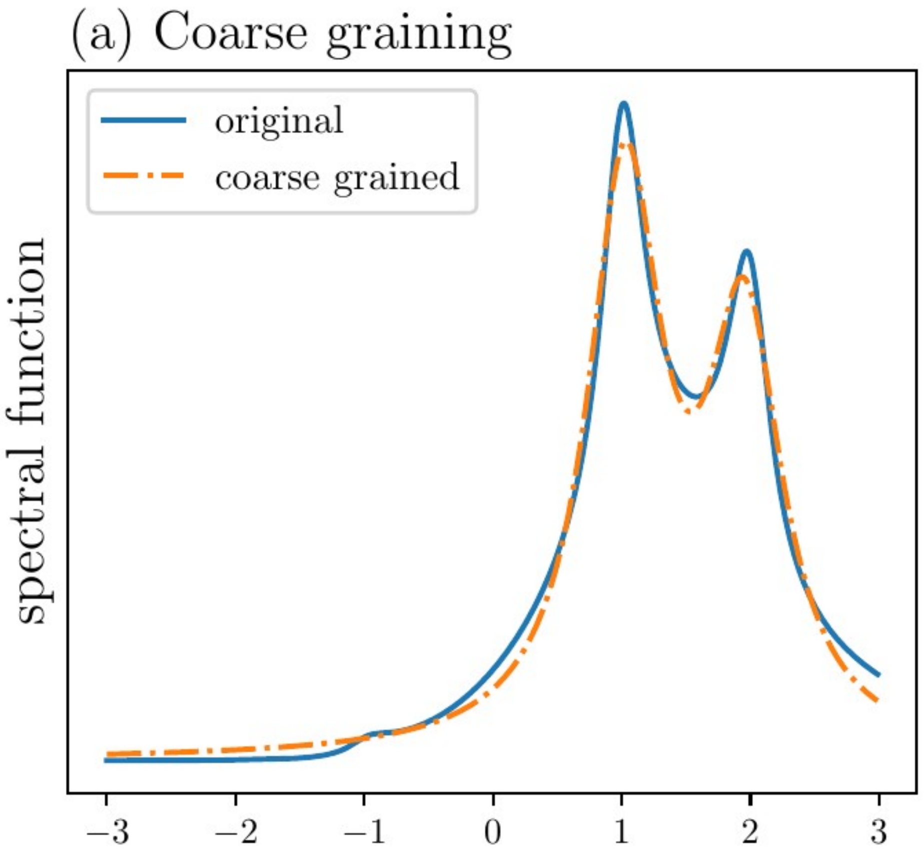
boson broadening

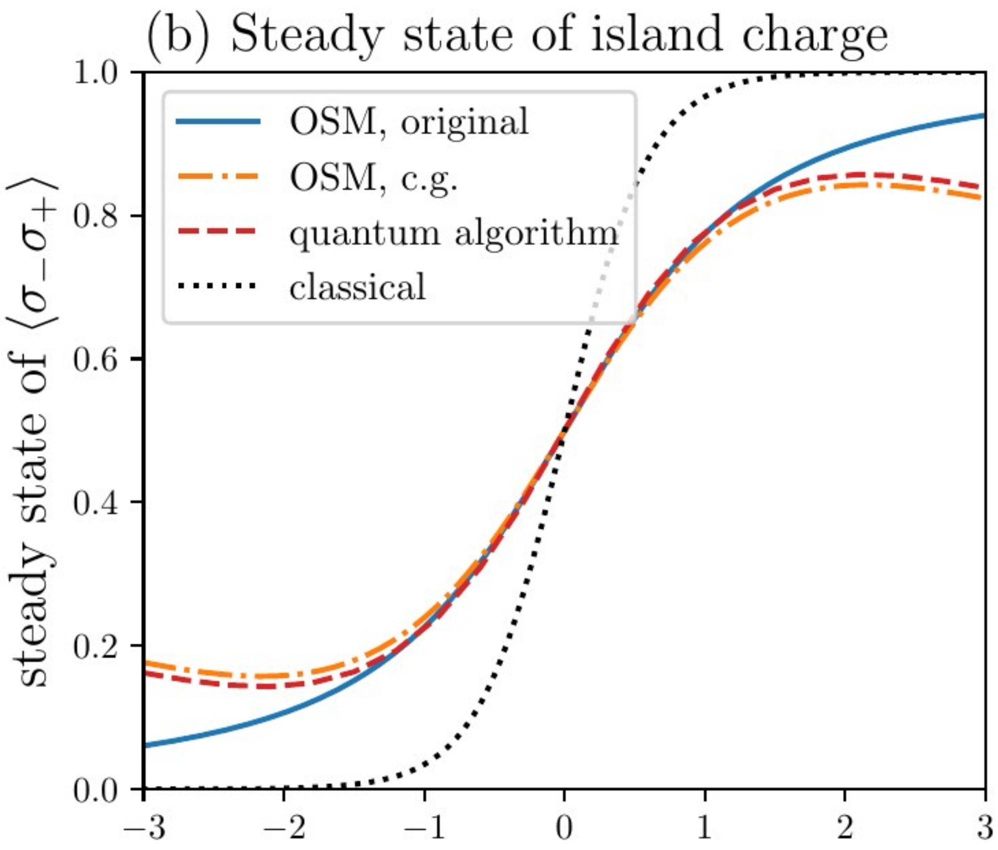
$$\kappa_i = \gamma_i^* = \gamma_i + 2\Gamma_i$$

Example results



- Finite gate fidelity due to incoherent error
- Bath qubit error is damping
- System qubit has connectivity to all bath qubits





Thank you!

This work was supported by the German Federal Ministry of Education and Research, through PhoQuant (13N16107), and QSolid (13N16155), and by the German Federal Ministry of Economic Affairs and Climate Action, through the PlanQK project (01MK20005H). This work was also supported by the European Union's Horizon 2020 program number 899561, AVaQus.

Gefördert durch:

Bundesministerium für Wirtschaft und Klimaschutz

aufgrund eines Beschlusses des Deutschen Bundestages

PlanQK

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung

QSOLID

Q-EXA

European Commission

Horizon 2020
European Union funding
for Research & Innovation

AVaQus

HQS QUANTUM SIMULATIONS

Thank you!