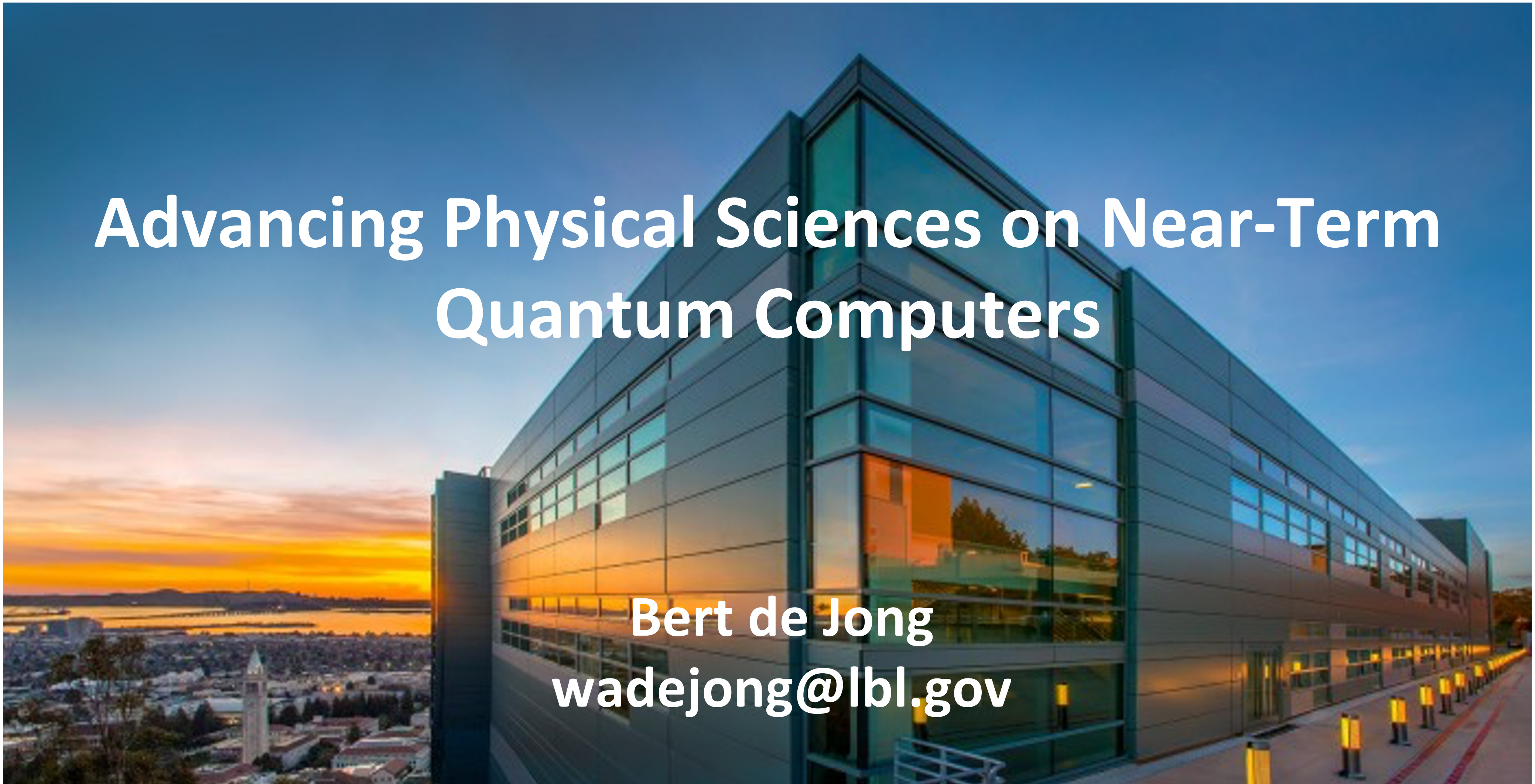


Advancing Physical Sciences on Near-Term Quantum Computers

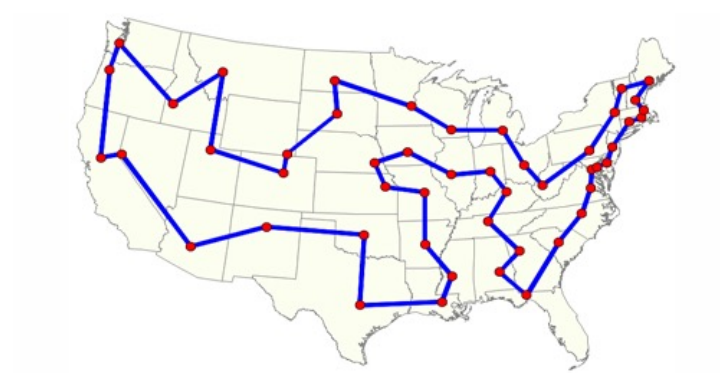
Bert de Jong
wadejong@lbl.gov

What makes quantum computing so exciting?

- **Speedups over classical computing**

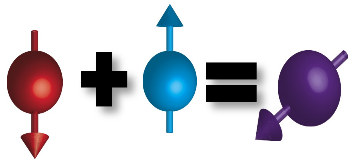
- **“Unbreakable” encryption protocols**

- **Quantum simulation**

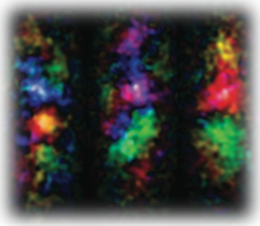


- **Efficient optimization algorithms**

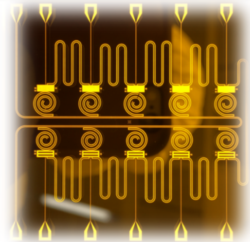
Quantum computing hardware technologies



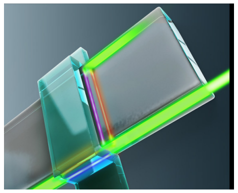
ELECTRONS
SPIN UP + SPIN DOWN



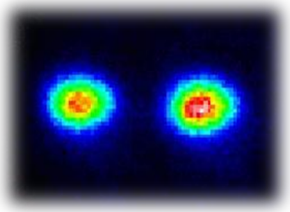
ATOMS



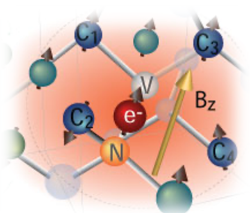
SUPER-
CONDUCTING



MAJORANA
QUASI-PARTICLE

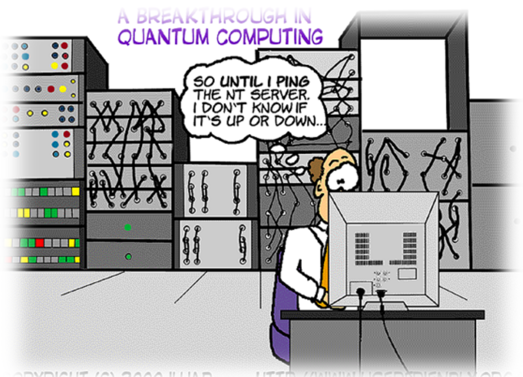


IONS



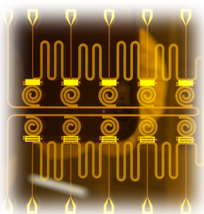
SOLID STATE
(spins)

D-WAVE

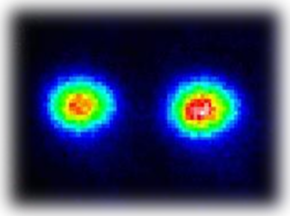


D-Wave The Quantum Computing Company™
IBM
Google
rigetti
IONQ
Microsoft
intel

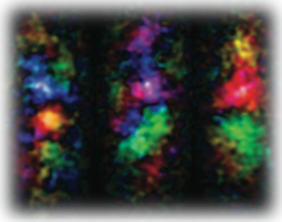
Many challenges with quantum hardware



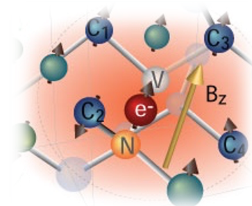
CIRCUITS



IONS



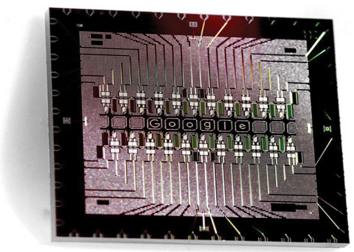
ATOMS

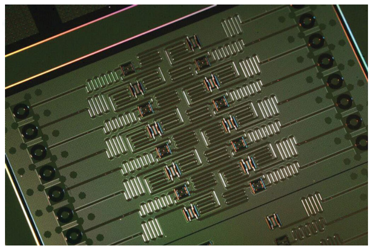


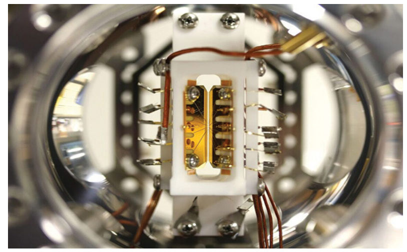
SOLID STATE

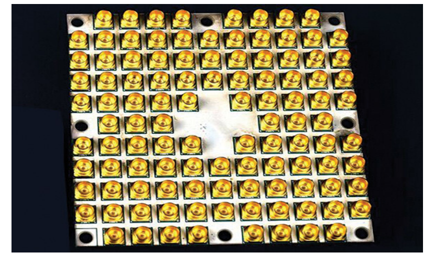
- **# of good qubits not yet enough for quantum advantage/science**
- **Coherence (available compute time) very short (10s-100s of ops)**
- **Noise and errors still pretty large**
- **Diverse technologies, each with its own instruction set**
- **Software tools and compilers are still in their infancy**

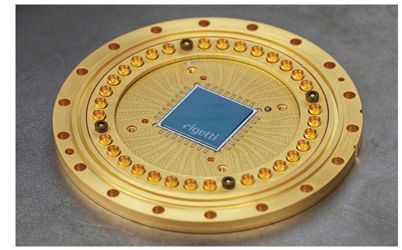
For example, gate sets in superconducting chips


Google

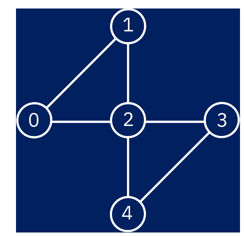

IBM


Rigetti


Intel


IonQ

- **Each chip has own native gate set**
  - Single qubit, usually rotations, and Hadamard
  - Two-qubit, usually CNOT, CZ (Google), SWAP
- **Each chip has a constrained topology**
  - Ring, array, mesh, bow-tie



- **Compilers needed to translate gate sets, do mapping**

Making quantum simulations feasible with NISQ

- **Requires minimizing number of qubits and depth of circuit**
  - Novel basis, using physical knowledge (symmetries, electron count, etc.) reduces qubit count
  - Localized Hamiltonians reduce number of interactions, reducing circuit depth and level of entanglement Interaction picture, qubitization

- **Using error mitigation strategies**

Towards useful quantum computing for science

Hardware technology

Scientific algorithms and software

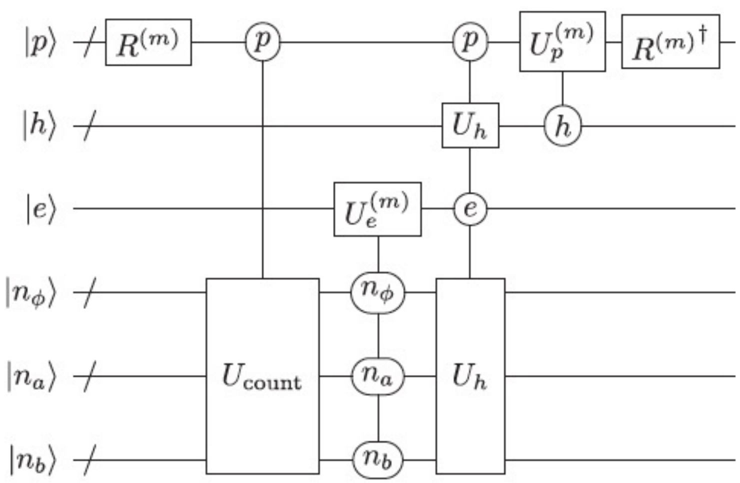
- Increasing qubit count
- Increasing lifetimes
- Increasing fidelity and reducing errors

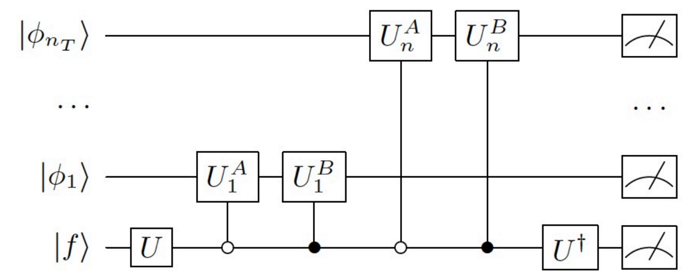
- Reducing qubit count
- Decreasing operation counts
- Incorporating error resiliency

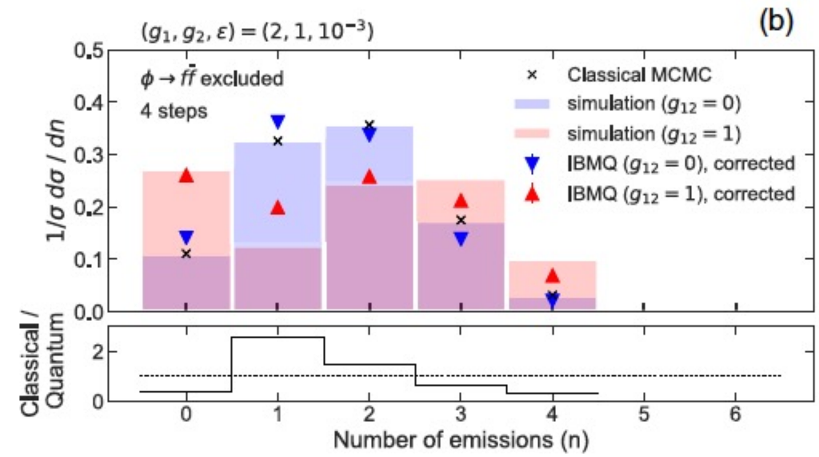
Parton showers and interfering trees

Fermions interacting with scalar boson

$$\mathcal{L} = \bar{f}_1(i\partial\!\!\!/ + m_1)f_1 + \bar{f}_2(i\partial\!\!\!/ + m_2)f_2 + (\partial_\mu\phi)^2 + g_1\bar{f}_1 f_1\phi + g_2\bar{f}_2 f_2\phi + g_{12}[\bar{f}_1 f_2 + \bar{f}_2 f_1]\phi.$$



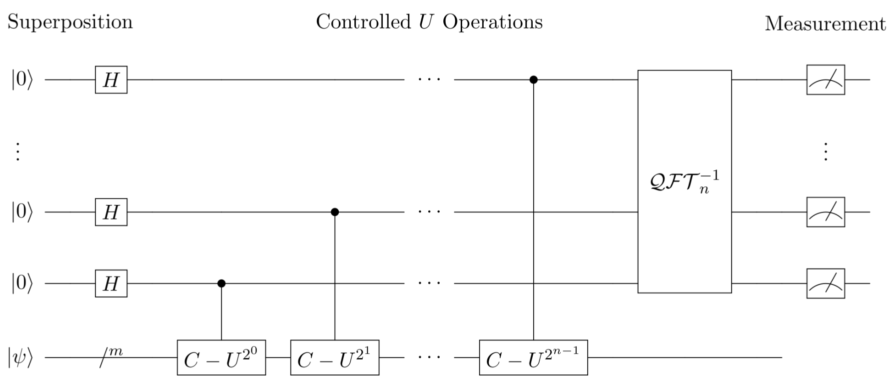




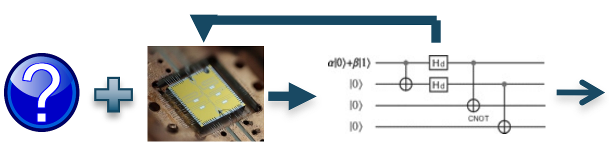
Provasoli, Nachman, Bauer, de Jong, Quant. Sci. Tech. 5, 035004 (2020)
Nachman, Provasoli, de Jong, Bauer, Phys. Rev. Lett. 126, 062001 (2021)

Two common algorithms for quantum simulations

Quantum Phase Estimation (QPE)

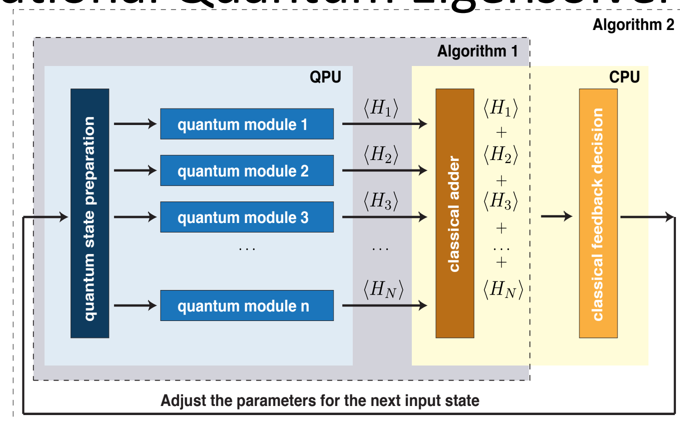


Prepare, evolve, FT and measure to find eigenvalue for eigenvector



42.02

Variational Quantum Eigensolver (VQE)

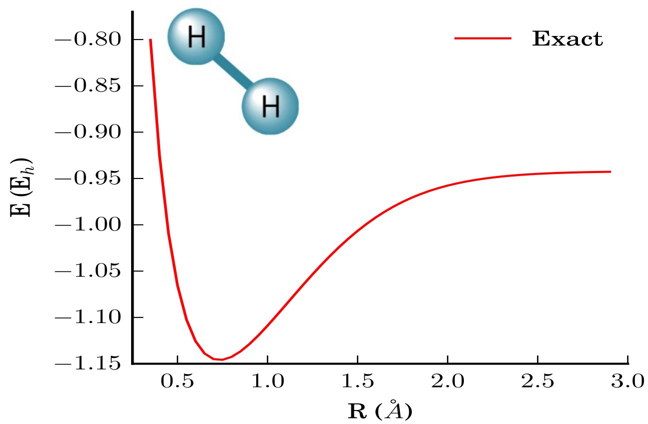


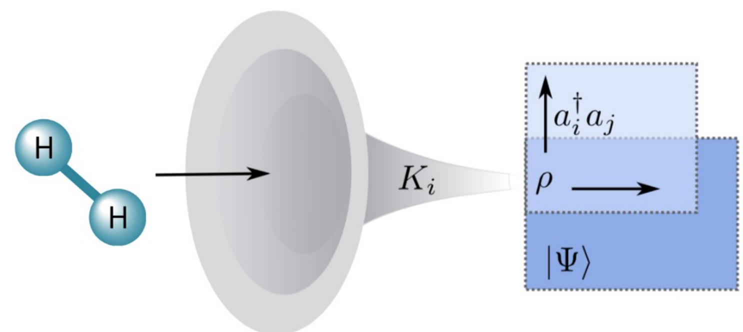
$$H = \sum_{i\alpha} g_i^{\alpha} \langle \sigma_\alpha^i \rangle + \frac{1}{2} \sum_{ij\alpha\beta} g_{ij}^{\alpha\beta} \langle \sigma_\alpha^i \sigma_\beta^j \rangle + \cdots$$

Only prepare and measure, do the rest classically

Quantum subspace expansion (QSE)

Quantum State on Quantum Device → Expand to Linear Response (LR) Subspace
Extra Quantum Measurements



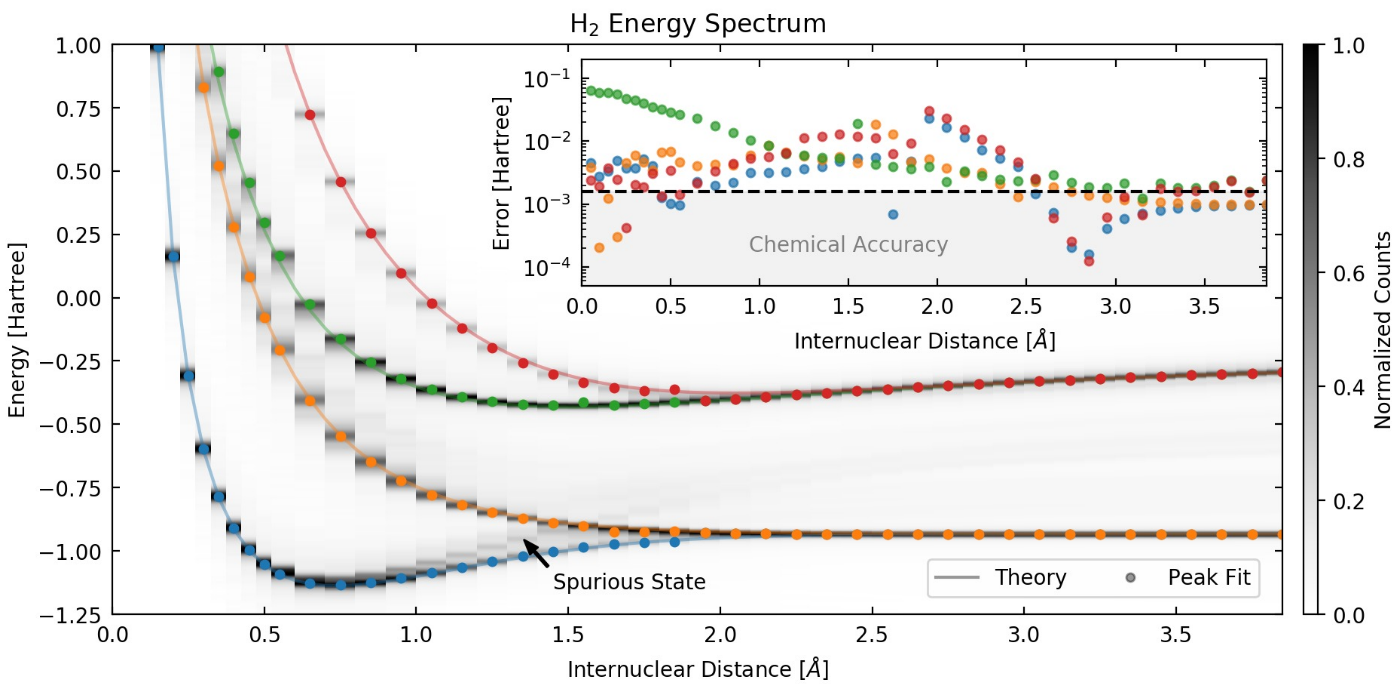


Excited State Energy and Properties ← Classical Generalized Eigenvalue Problem

$$HC = SCE$$

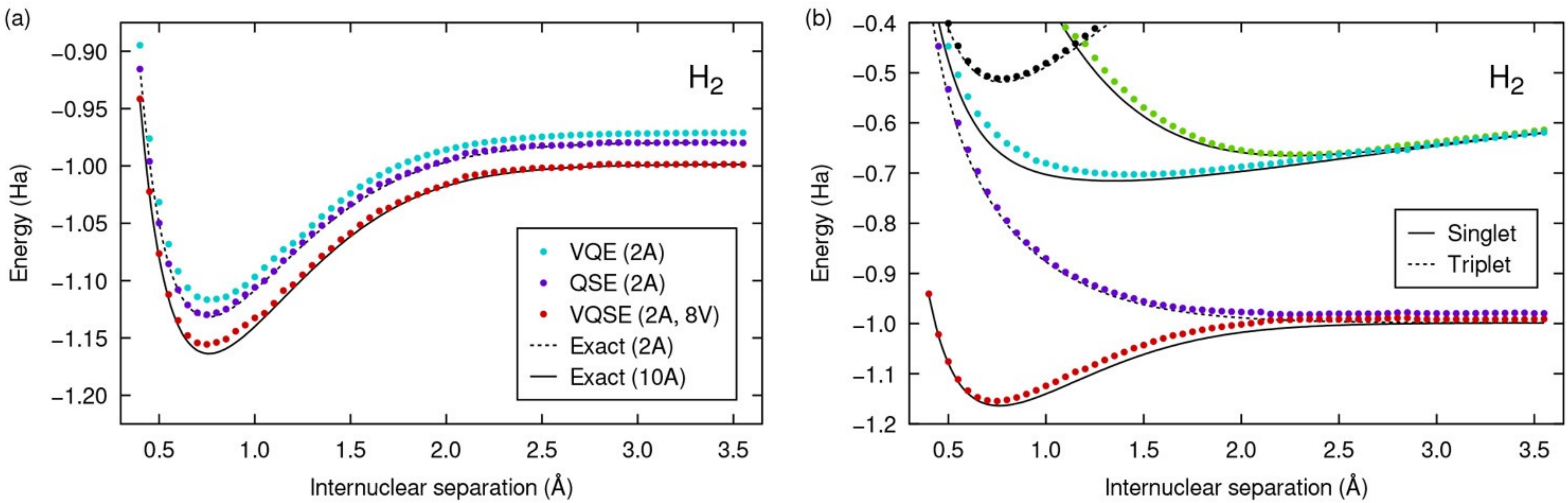
McClean, Schwartz, Carter, de Jong, Phys. Rev. A 95 (4), 042308 (2017)

Demonstrated end-to-end simulation on Berkeley hardware



Colless, J.I., Ramasesh, V.V., Dahlen, D., Blok, M.S., McClean, J.R., Carter, J., de Jong, W.A., Siddiqi, I. - Phys. Rev. X 8, 011021 (2018)

Solving noisy generalized eigensolver problem a challenge



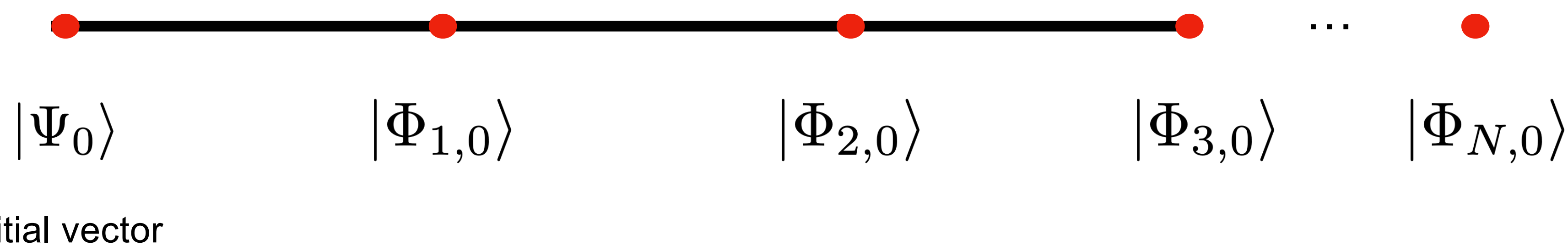
Special care needs to be taken in general eigensolver due to noise in data!

$$HC = SCE$$

Urbanek, Van Beeumen, et al., J. Chem. Theory Comput. 16, 5425 (2020)

Building on QSE:
Real-time evolution for eigenvalue extraction

Real time evolution to generate a basis of expansion states: $|\Phi_{j,0}\rangle = e^{-iHt_j}|\Psi_0\rangle$



Use as a basis to solve: $H\Psi = ES\Psi$

$$H_{i,j} = \langle \Phi_i | H | \Phi_j \rangle$$
$$S_{i,j} = \langle \Phi_i | \Phi_j \rangle$$

Possible to extract eigenstates by the cancellation of phases of components of the initial vector.

Promising because unlike imaginary, real time evolution is native to quantum computing.

Klymko, Mejuto-Zaera, de Jong, Tubman et al., arXiv:2103.08563

Variational Quantum Phase Estimation (VQPE)

Original generalized eigenvalue equation:

$$H\mathbf{c} = ES\mathbf{c}$$

→

Unitary form:

$$U(\Delta t)\boldsymbol{c} = e^{-iE\Delta t}S\boldsymbol{c}$$

$$U(\Delta t)_{j,k} = \langle\Psi_0|e^{-iH(\Delta t+t_k-t_j)}|\Psi_0\rangle = S_{j,k+1} = S_{j-1,k}$$

~~$H_{i,j} = \langle\Phi_i|H|\Phi_j\rangle$~~

$$S_{i,j} = \langle\Phi_i|\Phi_j\rangle$$

Autocorrelation Function

Toeplitz structure!

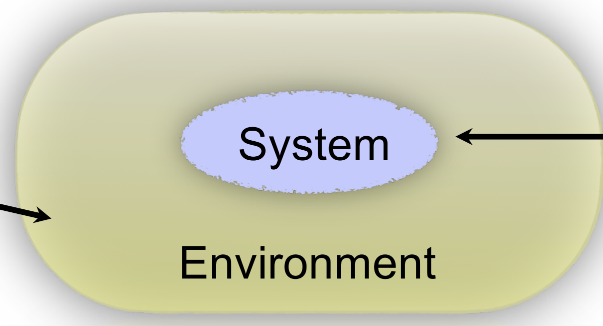
Toeplitz structure means that we only need a ***linear*** number of measurements instead of quadratic

Approach allows extraction of the maximal number of excited states!

Klymko, Mejuto-Zaera, Tubman et al., on arXiv:2103.08563

Towards engineering open quantum systems

Environment with
large degree of freedom



System
eigenstate populations
Boltzmann distributed

$$\rho_{th} = \frac{e^{-\beta H}}{Tr(e^{-\beta H})}$$

In limit $T \to 0$:
system ground state

Can we engineer quantum simulators to generate thermal states for arbitrary systems?

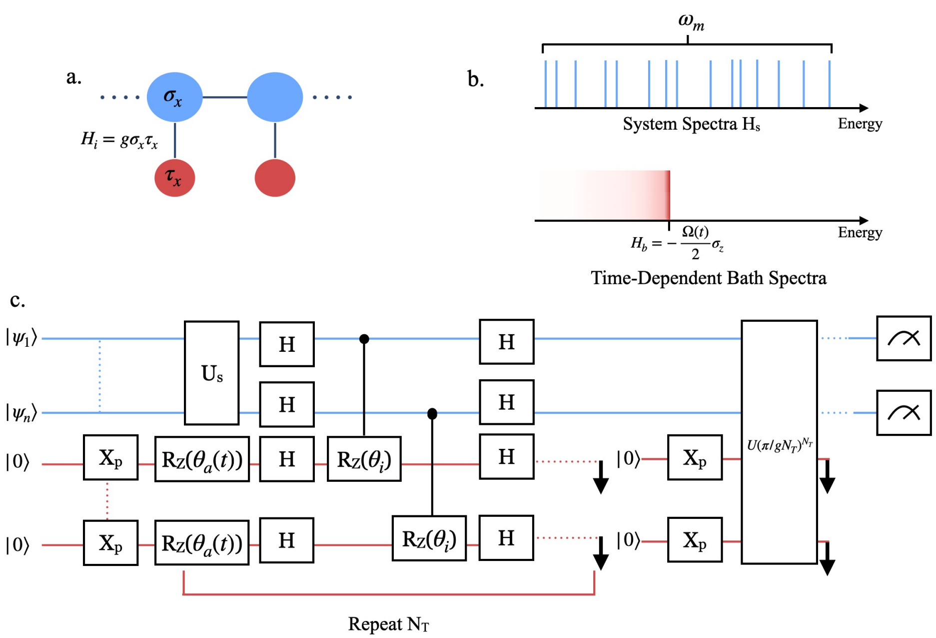
Some challenges:

- Simulating environment with many DOF
- Guaranteeing unique, steady-states (the thermal state)
- Control over bath parameters
- Thermalizing within coherence time

Metcalf, Moussa, de Jong, Sarovar
Phys. Rev. Research **2**, 023214 (2020)

Key: Mimic a macroscopic reservoir in a controllable manner

Quantum Markov Chain Monte Carlo with Driven Dissipative Dynamics on Quantum Computers



a) Principal qubits (blue) locally connected to ancilla qubits (red). b) Time-dependent ancilla frequency combs the system energy spectra and resonantly exchanges energy with different energy transitions in the system at different times c) Quantum circuit to implement the interaction cycle dynamical map

Scientific Achievement

A team of researchers led by Berkeley Lab developed a quantum algorithm to sample from Boltzmann distributions on quantum computers by engineering open-quantum system dynamics.

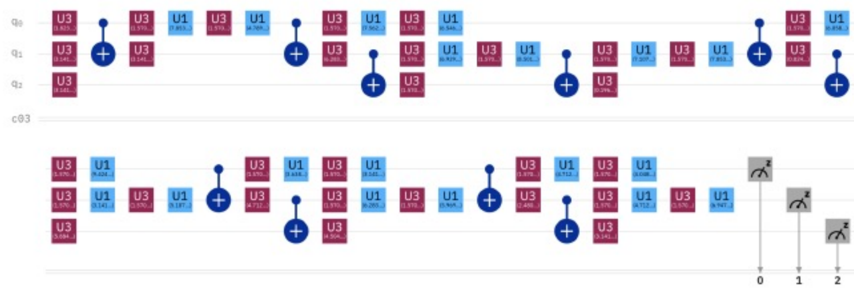
Significance and Impact

Our algorithm is designed to prepare robust, thermal states on quantum computers enabling finite-temperature simulations on quantum computers relevant to chemistry, materials and machine learning quantum applications.

Metcalf, Stone, Klymko, Kemper, Sarovar, de Jong - arXiv:2103.03207

Open quantum systems in heavy-ion collisions

Two-level system of heavy quark-antiquark pair (H_s) interacting with quark-gluon plasma (H_e) via interaction Hi with strength g.

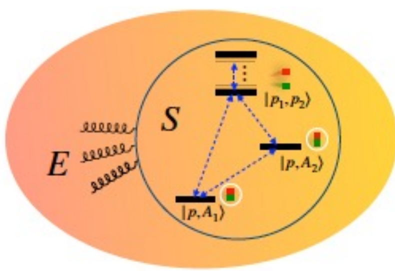


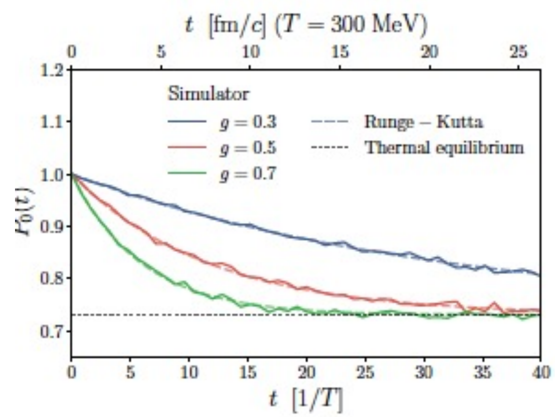
$$H_S = H_{S0} = -\frac{\Delta E}{2} Z$$

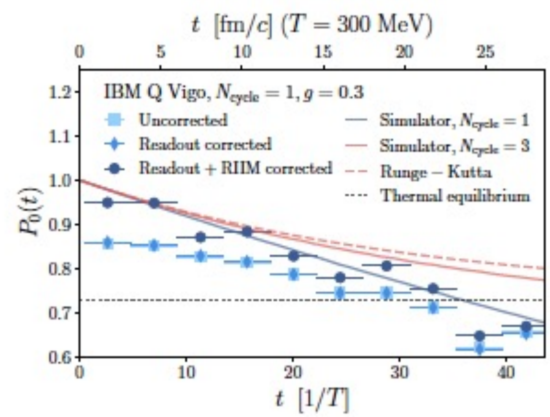
$$H_E = \int \mathrm{d}^3x \left[\frac{1}{2}\Pi^2 + \frac{1}{2}(\nabla\phi)^2 + \frac{1}{2}m^2\phi^2 + \frac{1}{4!}\lambda\phi^4\right]$$

$$H_I = gX \otimes \phi(x=0)\,,$$

$$L_j = \frac{\sqrt{\Gamma_j}}{2}(X - (-1)^j iY)\,,$$





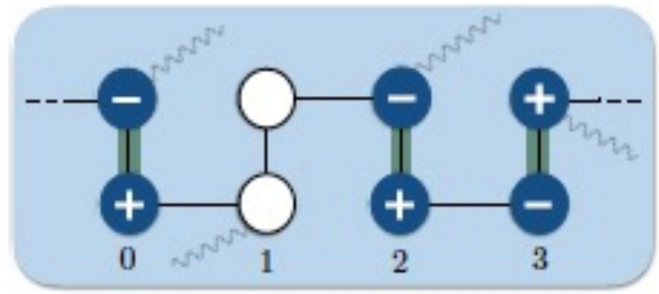


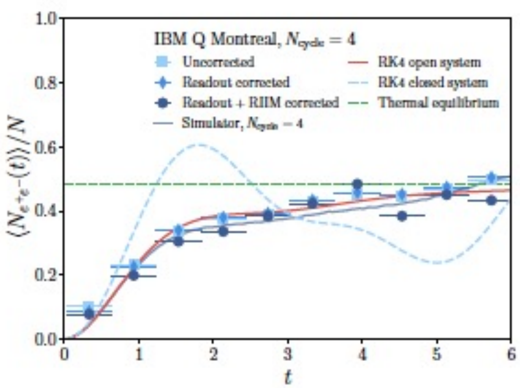
de Jong, Metcalf, Mulligan, Ploskon, Ringer, Yao, Phys. Rev. D 104, 051501 (2021)

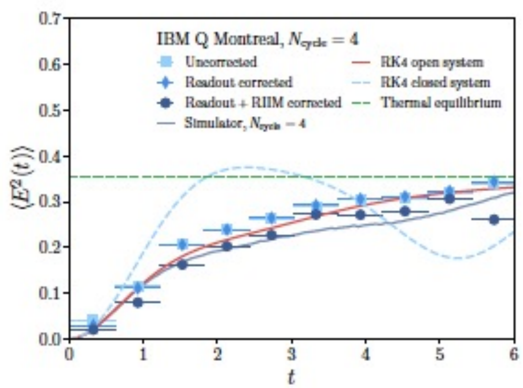
Non-equilibrium dynamics and thermalization in the Schwinger model

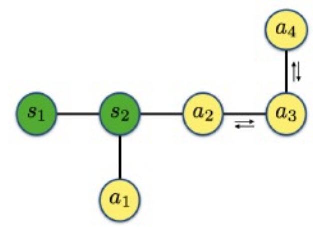
Fermion sites, uncopied or with electron/positron with electric field between fermions (green) and interaction with environment.

Demonstration below on 2 fermions sites an 1 qubit representing the environment.







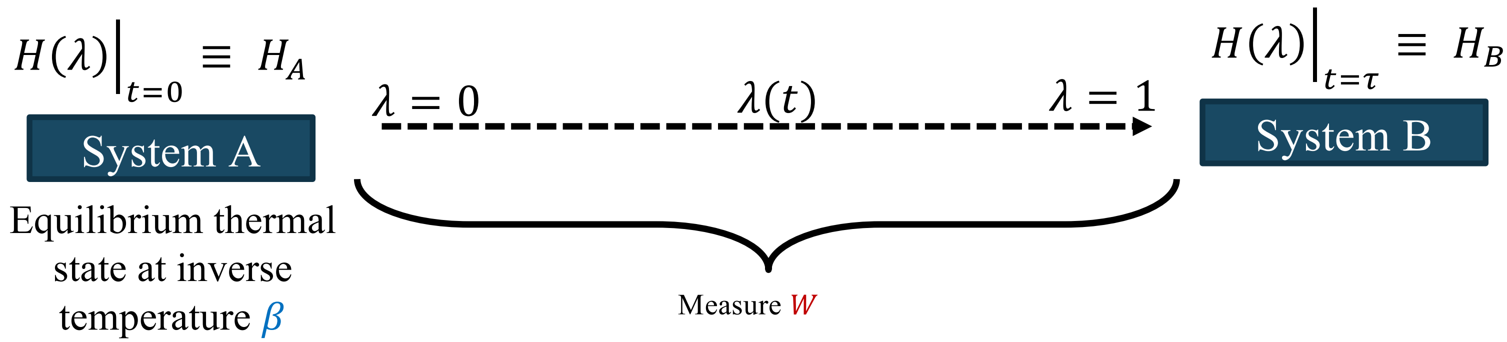


Qubit use, system in green, with 4 qubits for environment for 4 cycles as mid-circuit reset was not available.

de Jong, Metcalf, Mulligan, Ploskon, Ringer, Yao, arXiv:2106.08394

Latest editions of quantum computing systems do allow mid-circuit reset!

Computing Free Energy with Jarzynski Equality



$W \geq \Delta F$ → $e^{-\beta \Delta F} = \langle e^{-\beta W} \rangle$

Second Law of Thermodynamics

Jarzynski equality

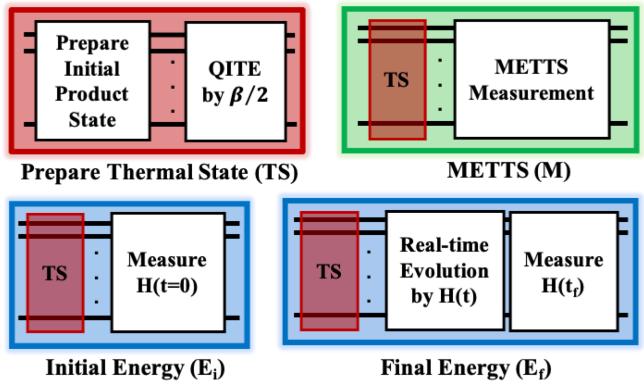
Challenges:

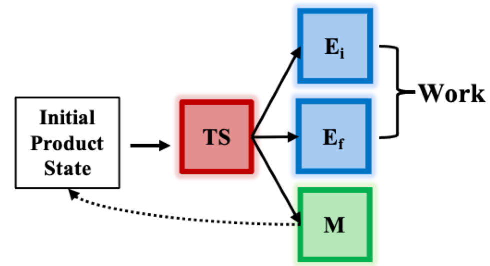
- Prepare thermal state on quantum computer
- Measure quantum work over trajectory

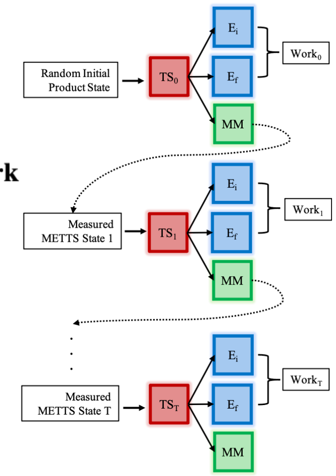
Bassman, Klymko, Liu, Tuman, de Jong – arXiv:2103.09846

Computing Free Energies with Fluctuation Relations on Quantum Computers

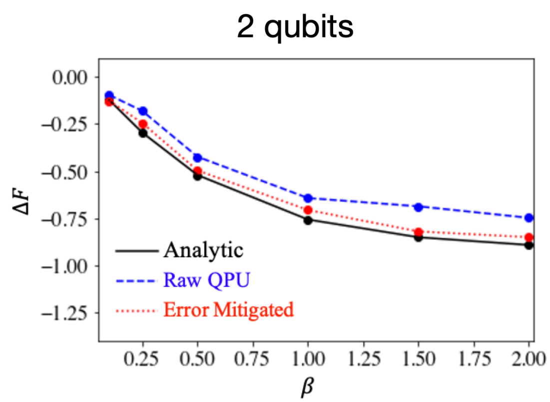
- Prepare thermal states on a quantum computer
- Measure work for a quantum trajectory

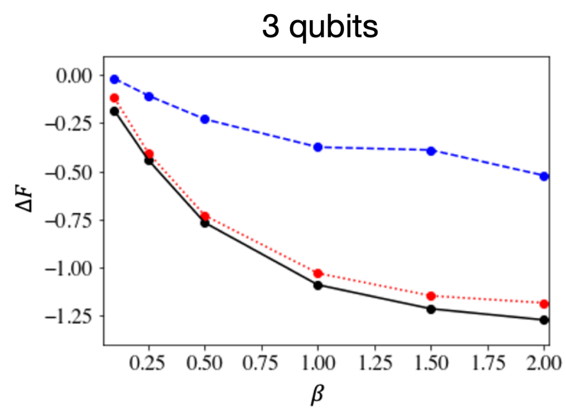






Hardware Results:





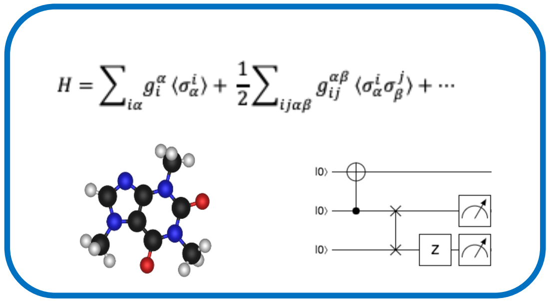
Bassman, Klymko, Tubman, de Jong - arXiv:2103.09846

METTS: Minimally Entangled Typical Thermal States

Programming a quantum computer

Scientist ⟷ *Hardware*

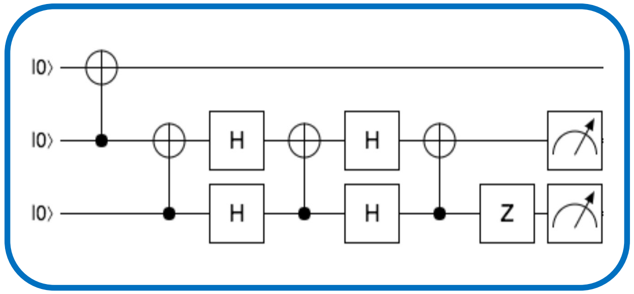
Science problem to Algorithm



High level interface

- Domain Specific Language
- Solvers (VQE, QPA, QITE)
- Algorithm specified in any gate set

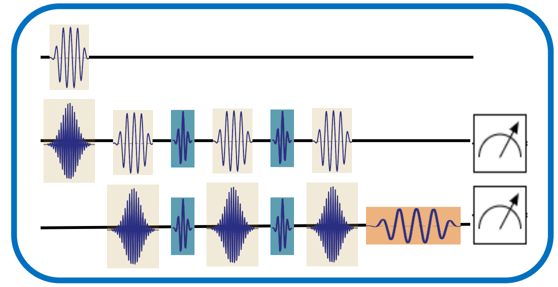
Compiled Quantum Algorithm



Translate to processor

- Arbitrary gates compiled into available gate set
- Processor connectivity and timing constraints enforced

Pulses output by AWG



Translation to hardware

- Define pulse parameters (shape, phase, sequence)
- Reset/feedback code applied by FPGAs

Adapted from Irfan Siddiqi, UC Berkeley

A far from complete list of software tools

- **Frameworks from most chip providers**

Provider	Framework	License	Cloud
IBM	QisKit	Minor restrictions	IBM Q-Experience
Google	Cirq	Open	
Rigetti	Forest / PyQuil	Restrictive	Rigetti QCS (beta)
Microsoft	LiQUi> / Q#	Minor restrictions	
D-Wave	qbsolv	Minor restrictions	D-Wave Leap
Xanadu	Strawberry Fields	Open	

- **Academia & startups target the above**
  - E.g. PyTKET (Cambridge Quantum), ProjectQ (ETH Zürich)
  - QuTiP (Academia, also RIKEN; http://qutip.org)

Quantum JIT (QJIT)

- **JIT for quantum kernels in qcor**
  - Flexibility to construct kernels based on runtime information
- **Kernels passed as strings, output function pointer to execute**
- **QJIT class - integration of …**
  - QCOR SyntaxHandler (map kernel to valid C++ code)
  - Clang LLVM IR CodeGen (map C++ code to LLVM Module)
  - LLVM JIT (extract function pointer from LLVM Module)
- **Extended to Python…**

```cpp
#include "qcor_jit.hpp"

int main() {

  // QJIT is the entry point to QCOR quantum kernel
  // just in time compilation
  QJIT qjit;

  // Define a quantum kernel string dynamically
  const auto kernel_src = R"#(__qpu__ void bell(qreg q) {
        using qcor::openqasm;
        h q[0];
        cx q[0], q[1];
        creg c[2];
        measure q -> c;
    })#";

  // Use the QJIT instance to compile this at runtime
  qjit.jit_compile(kernel_src);

  // Now, one can get the compiled kernel as a
  // functor to execute, must provide the kernel
  // argument types as template parameters
  auto bell_functor = qjit.get_kernel<qreg>("bell");

  // Allocate some qubits and run the kernel functor
  auto q = qalloc(2);
  bell_functor(q);
  q.print();

  // Or, one can call the QJIT invoke method
  // with the name of the kernel function and
  // the necessary function arguments.
  auto r = qalloc(2);
  qjit.invoke("bell", r);
  r.print();
```

Quantum JIT (QJIT)

C++

```cpp
#include "qcor_qsim.hpp"

// Define a fixed ansatz as a QCOR kernel
__qpu__ void ansatz(qreg q, double theta) {
  X(q[0]);
  auto exponent_op = X(0) * Y(1) - Y(0) * X(1);
  exp_i_theta(q, theta, exponent_op);
}

int main(int argc, char **argv) {
  // Create the Deuteron Hamiltonian
  auto H = 5.907 - 2.1433 * X(0) * X(1) - 2.143 * Y(0) * Y(1) + 0.21829 *
Z(0) -
           6.125 * Z(1);
  const auto num_qubits = 2;
  const auto num_params = 1;
  auto problemModel =
      qsim::ModelBuilder::createModel(ansatz, H, num_qubits, num_params);
  auto optimizer = createOptimizer("nlopt");
  // Instantiate a VQE workflow with the nlopt optimizer
  auto workflow = qsim::getWorkflow("vqe", {{"optimizer", optimizer}});

  // Result should contain the ground-state energy along with the optimal
  // parameters.
  auto result = workflow->execute(problemModel);

  const auto energy = result.get<double>("energy");
  std::cout << "Ground-state energy = " << energy << "\n";
  return 0;
}
```

Python

```python
from qcor import *

# Define the deuteron hamiltonian
H = -2.1433 * X(0) * X(1) - 2.1433 * \
    Y(0) * Y(1) + .21829 * Z(0) - 6.125 * Z(1) + 5.907

# Define the quantum kernel by providing a
# python function that is annotated with qjit for
# quantum just in time compilation
@qjit
def ansatz(q : qreg, theta : float):
    X(q[0])
    Ry(q[1], theta)
    CX(q[1], q[0])

# Create the problem model, provide the state
# prep circuit, Hamiltonian and note how many qubits
# and variational parameters
num_params = 1
problemModel = qsim.ModelBuilder.createModel(ansatz, H,
num_params)

# Create the NLOpt derivative free optimizer
optimizer = createOptimizer('nlopt')

# Create the VQE workflow
workflow = qsim.getWorkflow('vqe', {'optimizer': optimizer})

# Execute and print the result
result = workflow.execute(problemModel)
energy = result['energy']
print(energy)
```

http://aide-ac.org/Software

Circuit Synthesis Language Extension in Python

Original C++ decompose extension

```
__qpu__ void ccnot(qreg q) {

  // set initial state to 111
  for (int i = 0; i < q.size(); i++) {
    X(q[i]);
  }

  // To program at the unitary matrix level,
  // invoke the decompose call, indicating which
  // buffer to target, can optionally provide decomposition
  // algorithm name and an optimizer.
  decompose {
    // Create the unitary matrix
    UnitaryMatrix ccnot_mat = UnitaryMatrix::Identity(8, 8);
    ccnot_mat(6, 6) = 0.0;
    ccnot_mat(7, 7) = 0.0;
    ccnot_mat(6, 7) = 1.0;
    ccnot_mat(7, 6) = 1.0;
  }
  (q);

  // Add some measures
  for (int i = 0; i < q.size(); i++) {
    Measure(q[i]);
  }
}
```

Now possible in Python

```
@qjit
def ccnot(q : qreg):
    # create 111
    for i in range(q.size()):
        X(q[i])

    with decompose(q) as ccnot:
        ccnot = np.eye(8)
        ccnot[6,6] = 0.0
        ccnot[7,7] = 0.0
        ccnot[6,7] = 1.0
        ccnot[7,6] = 1.0

    # CCNOT should produce 110 (lsb)
    for i in range(q.size()):
        Measure(q[i])
```

Can leverage Numpy

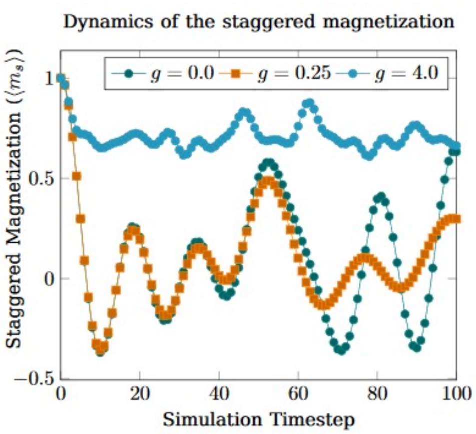
```
@qjit
def all_x(q : qreg):
    with decompose(q) as x_kron:
        sx = np.array([[0, 1],[1, 0]])
        x_kron = np.kron(np.kron(sx,sx),sx)

    for i in range(q.size()):
        Measure(q[i])
```

Composable Programming of Hybrid Workflows for Quantum Simulation

```
// AF Heisenberg model
auto problemModel = ModelFactory::createModel(
    "Heisenberg", {{"Jx", 1.0},
                  {"Jy", 1.0},
                  // Jz == g parameter
                  {"Jz", g},
                  // No external field
                  {"h_ext", 0.0},
                  {"num_spins", n_spins},
                  {"initial_spins",
                  initial_spins},
                  {"observable",
                  "staggered_magnetization"}});
// Time-dependent simulation workflow
auto workflow = getWorkflow("td-evolution",
                {{"dt", dt},
                {"steps", n_steps}});
// Execute the workflow
auto result = workflow->execute(problemModel);
```

Defining the antiferromagnetic (AF) Heisenberg problem model and simulating its dynamics with Quantum Simulation Modeling (QuaSiMo) library.



Results for the staggered magnetization of the AF Heisenberg model using quantum dynamics simulations with the QuaSiMo library

Scientific Achievement

We developed a composable design scheme for quantum simulation applications using the QCOR hardware-agnostic programming language into the QuaSiMo library.

Significance and Impact

The QuaSiMo library enables rapid synthesis of hybrid algorithms and workflows using a common, reusable methods and data structures for quantum simulation applications.

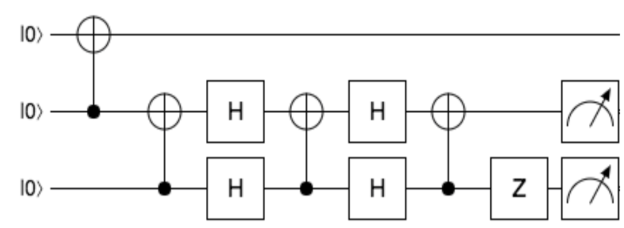
T. Nguyen, L. Bassman, D. Lyakh, A. McCaskey, V. Leyton-Ortega, R. Pooser, W. Elwasif, T. S. Humble and W. A. de Jong, “Composable Programming of Hybrid Workflows for Quantum Simulation,”in press arXiv:2101.08151 (2021)

Compiling…General Synthesis Problem

Unitary

$$\frac{1}{\sqrt{2^3}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \omega^{10} & \omega^{12} & \omega^{14} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} & \omega^{15} & \omega^{18} & \omega^{21} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} & \omega^{20} & \omega^{24} & \omega^{28} \\ 1 & \omega^5 & \omega^{10} & \omega^{15} & \omega^{20} & \omega^{25} & \omega^{30} & \omega^{35} \\ 1 & \omega^6 & \omega^{12} & \omega^{18} & \omega^{24} & \omega^{30} & \omega^{36} & \omega^{42} \\ 1 & \omega^7 & \omega^{14} & \omega^{21} & \omega^{28} & \omega^{35} & \omega^{42} & \omega^{49} \end{bmatrix}$$

Circuit



Quantum Compilation: *Given unitary U, find decomposition in terms of gates G from a (universal) fixed gate set*

Enables:

- Algorithm discovery
- Gate set and hardware exploration
- Global circuit optimization

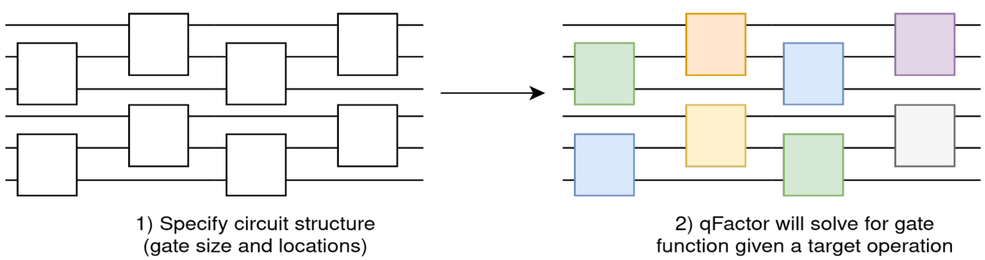
Berkeley Quantum Synthesis Toolkit

BQSKit

QSearch – Developed by Advanced Quantum Testbed

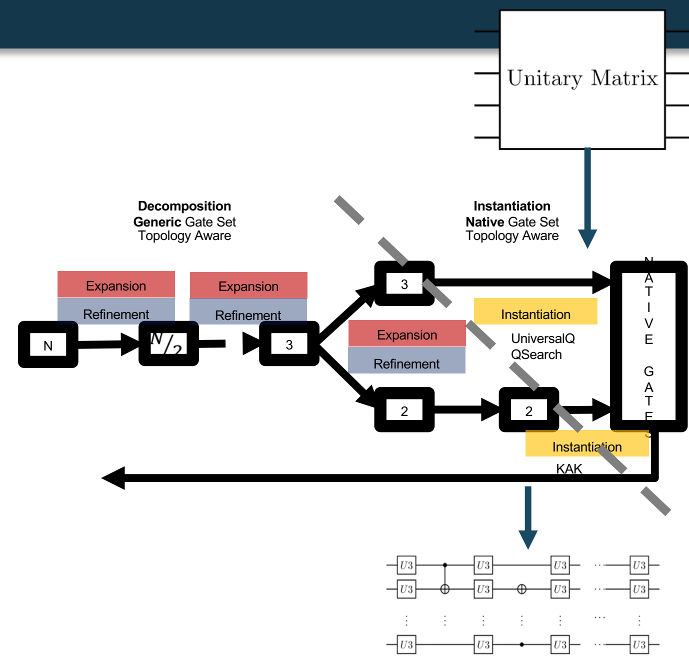
QFAST – Developed by QAT4Chem and AIDE-QC

Qfactor – Collaboration LANL (Cincio) and LBNL (Younis)



https://github.com/BQSKit

QFAST: Conflating Search with Numerical Optimization in Quantum Synthesis



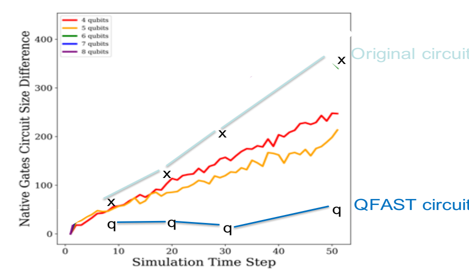
Scientific Achievement

We show how to increase scalability of "optimal" quantum synthesis by replacing search based algorithms with one step of numerical optimization

Significance and Impact

NISQ hardware requires short depth circuits, compilers have limits. QFAST can produce short circuits, hierarchical approach enables both scalability and retargetability/portability. QFAST is topology aware.

1. QFAST reduces depth 6.3X average, up to 30X
2. 10x shorter circuits than state-of-the-art



Ed Younis (LBL), Koushik Sen (UCB), Kathy Yelick (LBL), Costin Iancu (LBL) - QCE21 Best Paper Award

BERKELEY LAB

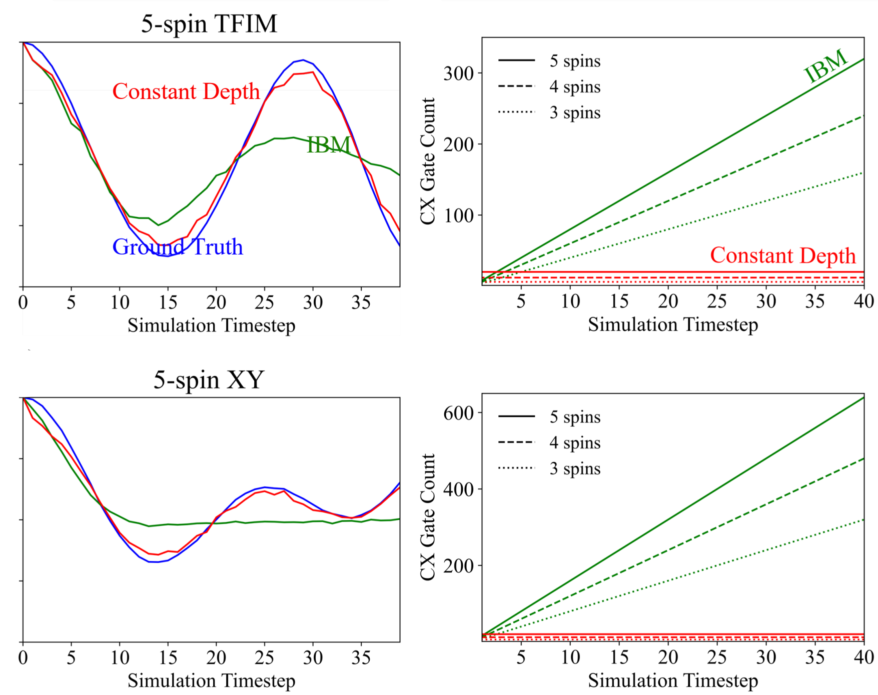
Constant-Depth Circuits for Dynamic Simulations on QC

Scientific Achievement

A method for generating circuits which are constant in depth with increasing time-step, thus enabling dynamic simulations on near-term quantum computers out to arbitrarily long simulation times.

Significance and Impact

High-fidelity simulation results for long-time dynamic simulations of quantum materials can be obtained on currently available quantum computers.



Comparison of simulation results and CX gate count for the TFIM and the XY model using the constant-depth circuits versus the IBM-compiled circuits.

Bassman, Van Beeumen, Younis, Smith, Iancu, de Jong
Mat. Theory, accepted (2021) - arXiv:2103.07429

QUEst: Robust Generation of Quantum Circuit Approximation Using Synthesis

Scientific Achievement

We provide a sound and scalable method for generating circuit approximations. Approximations significantly reduce circuit depth, while providing same output quality. On NISQ, approximations improve output quality.

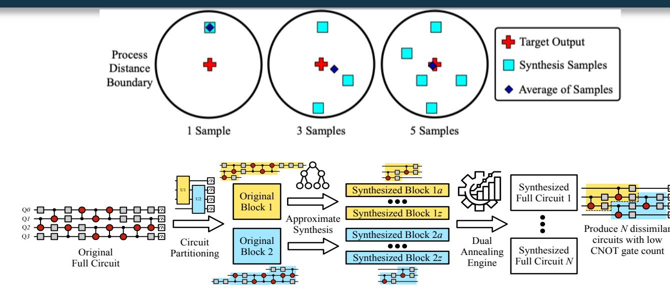
Significance and Impact

Resource efficiency is an important measure of circuit performance. Our technique can be directly used for circuit optimization in NISQ and fault-tolerant quantum computing. In NISQ, we provide additional capability for very good error mitigation.

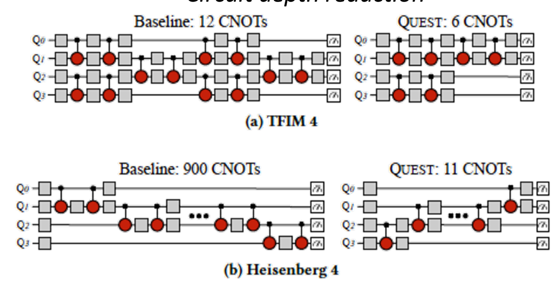
- We show 30%-80% depth reduction on many important algorithms
- We show fidelity improvements of 30% on noisy systems, independent of noise level
- The program output is accurate enough for science purposes
- Although computationally intensive, the technique scales up to high qubit counts (up 128 qubits demonstrated)

Patel, Younis, Bassman, Tiwari, de Jong, Iancu
ASPLOS 2022 - https://arxiv.org/abs/2108.12714

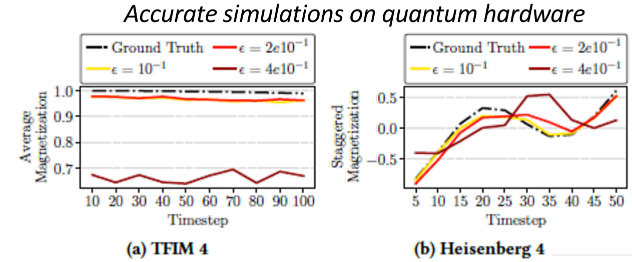
Sampling improves fidelity



Circuit depth reduction



Accurate simulations on quantum hardware



BERKELEY LAB

⟨AIDE|QC⟩

Northeastern University

AIDE-QC software stack available

- Binary installs for various platforms
- Extensive user and developer guide with examples

  http://aide-ac.org/Software

  https://aide-qc.github.io/deploy

Qubit errors due to relaxation and decoherence

T_1: relaxation, dampening

- Environment exchanges energy with the qubit, mixing the two states by stimulated emission or absorption
- Important during read-out

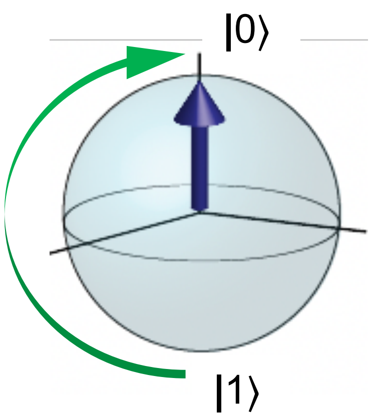
- **Intuitively time to decay from |1⟩ to |0⟩**

T_2: dephasing

- Environment creates loss of phase memory by smearing energy levels, changing phase velocity
- Important during "computation", bounds circuit depth (number of consecutive gates)

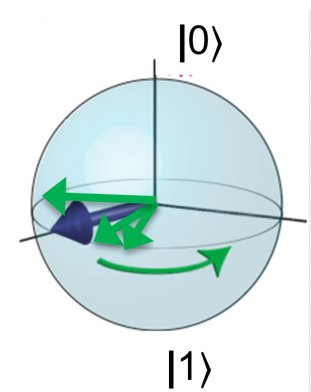
- **Intuitively time for ϕ to get imprecise**

$T_1 > T_2$

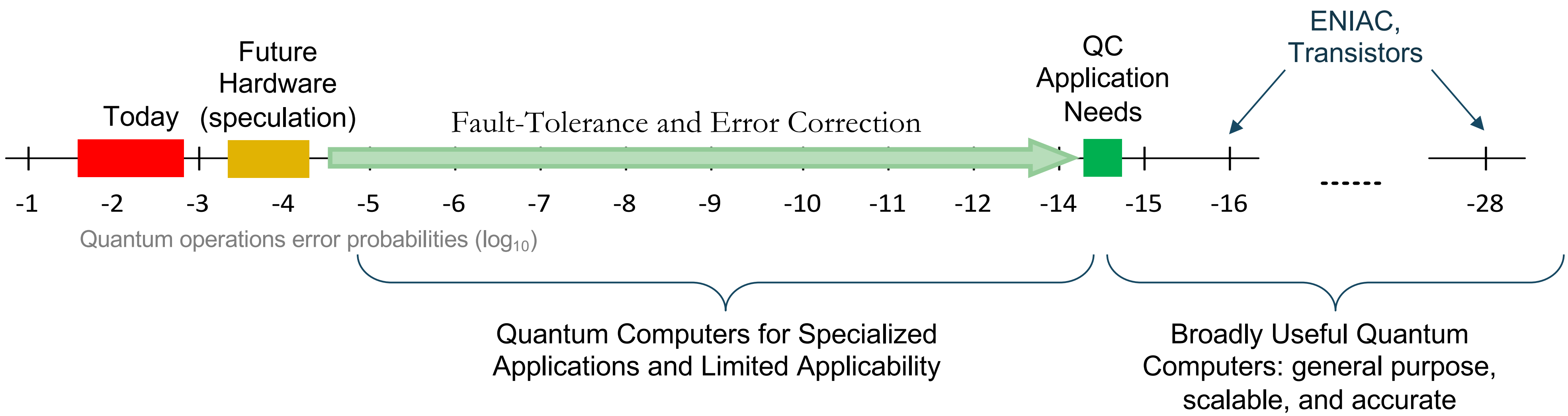


$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$$

These are not cut-off times, but "half-lives." Decay is *continuous*.



Noisy intermediate-scale quantum devices



- **Right now quantum computing is still a *physics experiment***
  - Noise is everywhere
  - Measurement errors

Correcting read-out error bias: Learning from noisy high-energy physics

"Iterative Bayesian Unfolding"

$$\theta_{ij} = \frac{\Pr(m_j|t_i) \cdot \Pr(t_i)}{\sum_i \Pr(m_j|t_i) \cdot \Pr(t_i)}$$

response matrix

regularization = number of iterations

$$\Pr_{k+1}(t_i) = \sum_j \theta_{ij} \Pr_k(t_i)$$

Nucl. Inst. Meth. A 362 (1995) 487

"Singular Value Decomposition (SVD) Unfolding"

$$R = USV^T$$

U, *V*, orthogonal, *S* diagonal & non-negative

$$d = U^T m \qquad z_i(\tau) = \frac{d_i}{s_i} \cdot \frac{s_i^2}{s_i^2 + \tau}$$

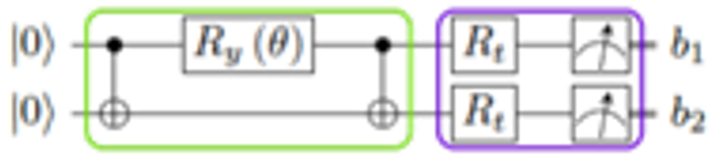
$$t = Vz$$

regularization parameter

Nucl. Inst. Meth. A 372 (1995) 469

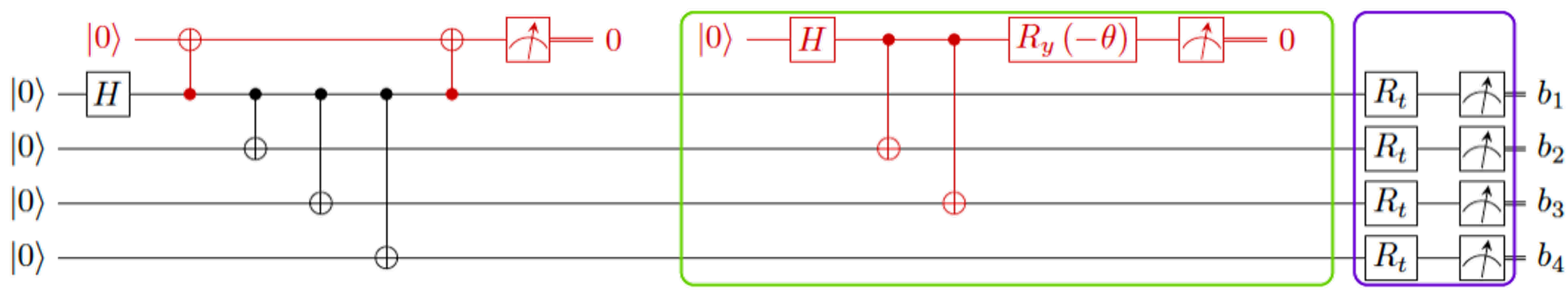
Regularized matrix inversion approaches

Building error detection/correction into algorithms



Single error detection

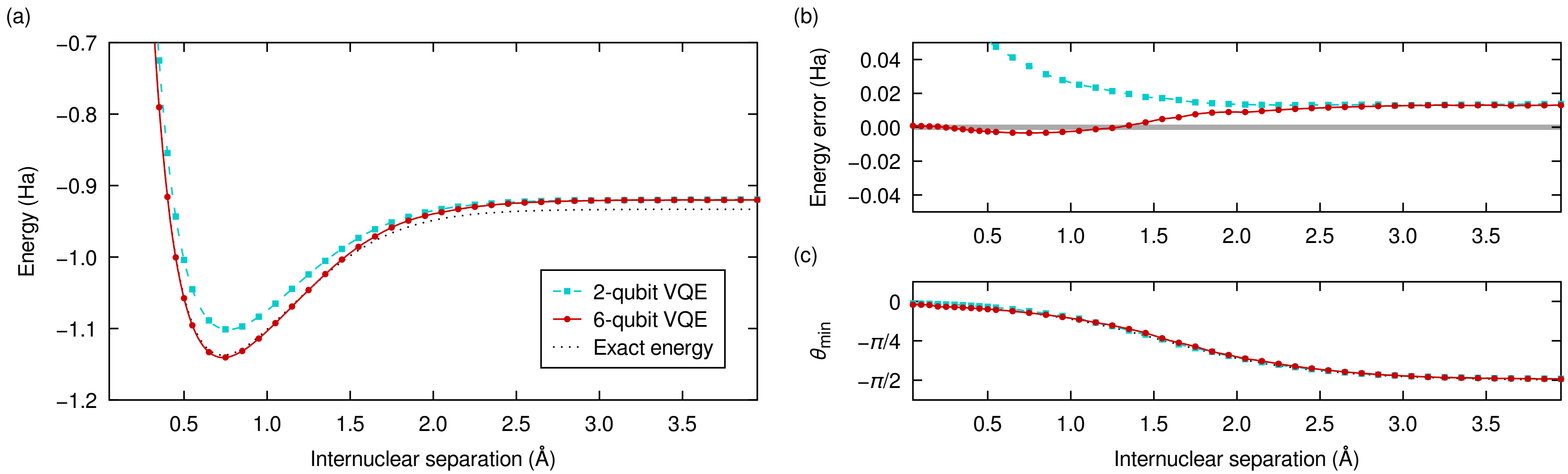
Magic states



[[4,2,2]] error correction code, with 2 logical qubits

Urbanek et al, Phys. Rev. A 102, 022427 (2020)

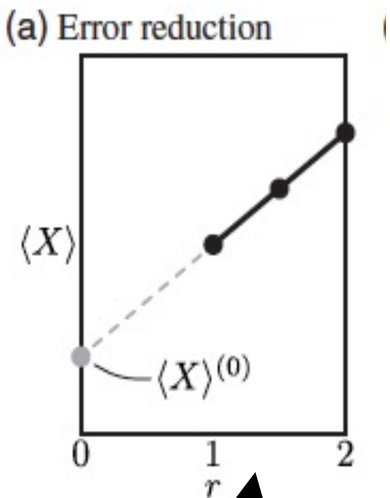
Error corrected circuits drastically improve results



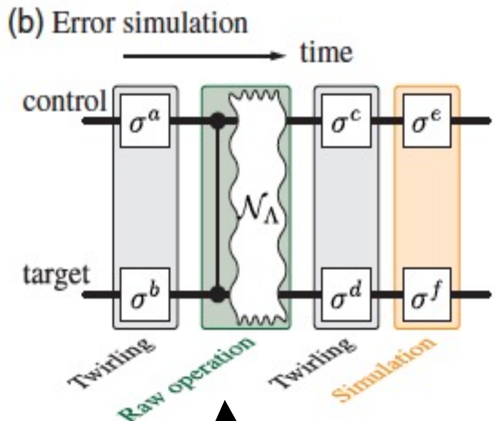
H_2 molecule on 2/6 qubits with minimal basis

Urbanek et al, Phys. Rev. A 102, 022427 (2020)

Tackling noise in quantum operations without error correction



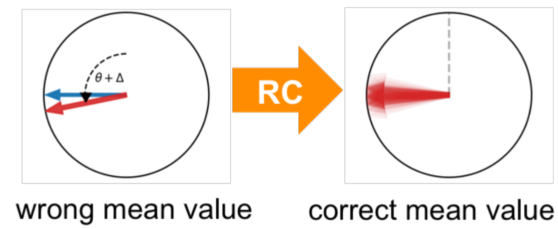
Adding error on purpose



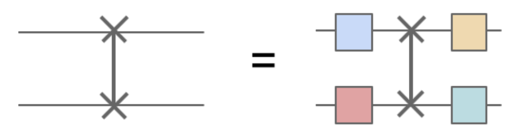
Converting non-stochastic to stochastic (randomized compiling)

Concept

- **Add gates that randomize signs of errors each run so *average* result is correct:**



- Implementation: Sandwich "hard" gates between certain random "easy" gates



Ying Li and Simon C. Benjamin - Phys. Rev. X 7, 021050 (2017)

Limits of error mitigation with linear extrapolation

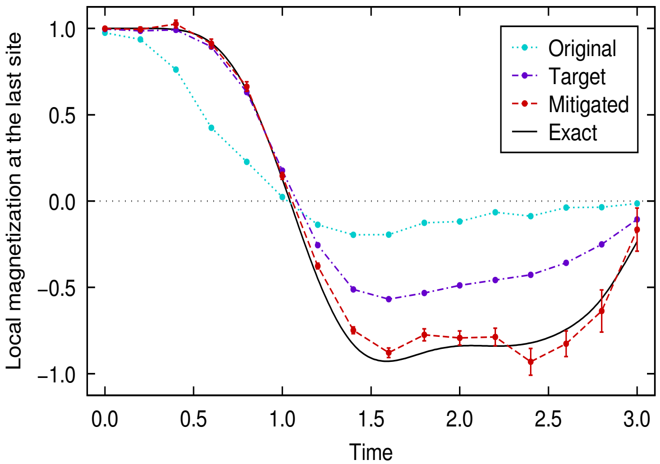
- **Primary challenge in circuits are CNOTs**
- **What if your circuit has 16 CNOTs…or more?**
  - Would require 16+32 and 16+32+32 CNOTs to add 1 and 2 layers of identities to extrapolate
- **Alternative is adding fewer identities (RIIM)**
  - Including fewer identities and using polynomial fits

$$\max\left[\delta, \epsilon^{n_{\max}}, \Delta_{\text{shot}}\right]$$

HEP QuantiSED

He et al., Phys. Rev. A 102, 012426 (2020)

Practical error-mitigation strategy for superconducting qubits



Effect of mitigation: Time evolution of a six-qubit magnetic model calculated without and with error mitigation.

Longest time = 220 CNOTs

Scientific Achievement

Design of a practical mitigation strategy for reducing errors and noise present in quantum computers based on superconducting qubits.

Significance and Impact

Combination of both existing and own mitigation approaches that address various error sources improves accuracy for quantum circuits with hundreds of gates.

Research Details

– *Readout error correction* mitigates readout errors occurring during a measurement.

– *Randomized compiling* mitigates coherent gate errors and converts them to incoherent noise.

– Noise correction with *noise estimation circuits* mitigates global depolarizing noise.

– *Error extrapolation* mitigates local depolarizing noise.

Urbanek, Nachman, Pascuzzi, He, Bauer, de Jong, Phys. Rev. Lett. 127, 270502 (2021)

Interdisciplinary team at LBNL

Physics

- Mekena Metcalf
- Miro Urbanek
- Lindsay Bassman
- Katie Klymko
- Ben Nachman
- Christian Bauer
- James Mulligan
- Matheusz Ploskon
- Felix Ringer
- Xiaojun Yao

Error mitigation

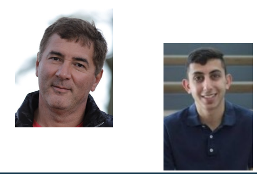
- Wim Lavrijsen
- Ben Nachman
- Christian Bauer
- Miro Urbanek

Mathematics

- Daan Camps
- Roel van Beeumen
- Julie Mueller

Compilers

- Costin Iancu
- Ed Younis



Acknowledgements

This work was supported by the DOE Office of Advanced Scientific Computing Research (ASCR) through the Quantum Algorithm Team and Quantum Testbed Pathfinder Program

This research used computing resources of the Oak Ridge Leadership Computing Facility through the INCITE program and the National Energy Research Scientific Computing Center

⟨AIDE|QC⟩

HEP QuantiSED

bq

qat4chem.lbl.gov

aide-qc.org

berkeleyquantum.org

thequantuminformationedge.org

FAR-QC

QAT4Chem

Bert de Jong

wadejong@lbl.gov

Berkeley Lab · Berkeley University of California · University of Toronto · Argonne National Laboratory