BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY
Office of Science

FABLE
Fast Approximate Block-Encodings

Daan Camps, Roel Van Beeumen

arXiv:2205.00081

September 20, 2022

IEEE Quantum Week 2022 – Broomfield, CO

NERSC

COMPUTING SCIENCES RESEARCH
LAWRENCE BERKELEY NATIONAL LABORATORY

Outline of the talk

FABLE: Quantum Circuits for Block-Encodings

01

Problem statement

02

Uniformly controlled rotations and FABLE circuits for real-valued and complex-valued matrices

03

Results on circuit compression and approximation

04

Examples on Hamiltonians and differential operators

05

Application: Preparation of thermal states

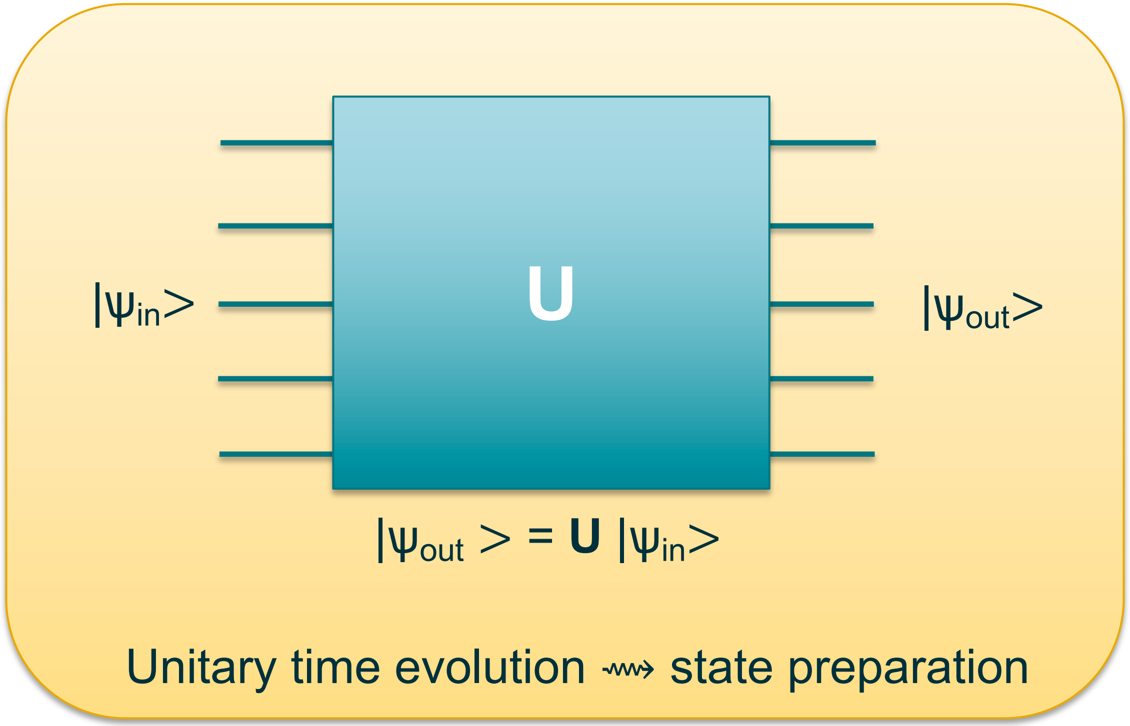
06

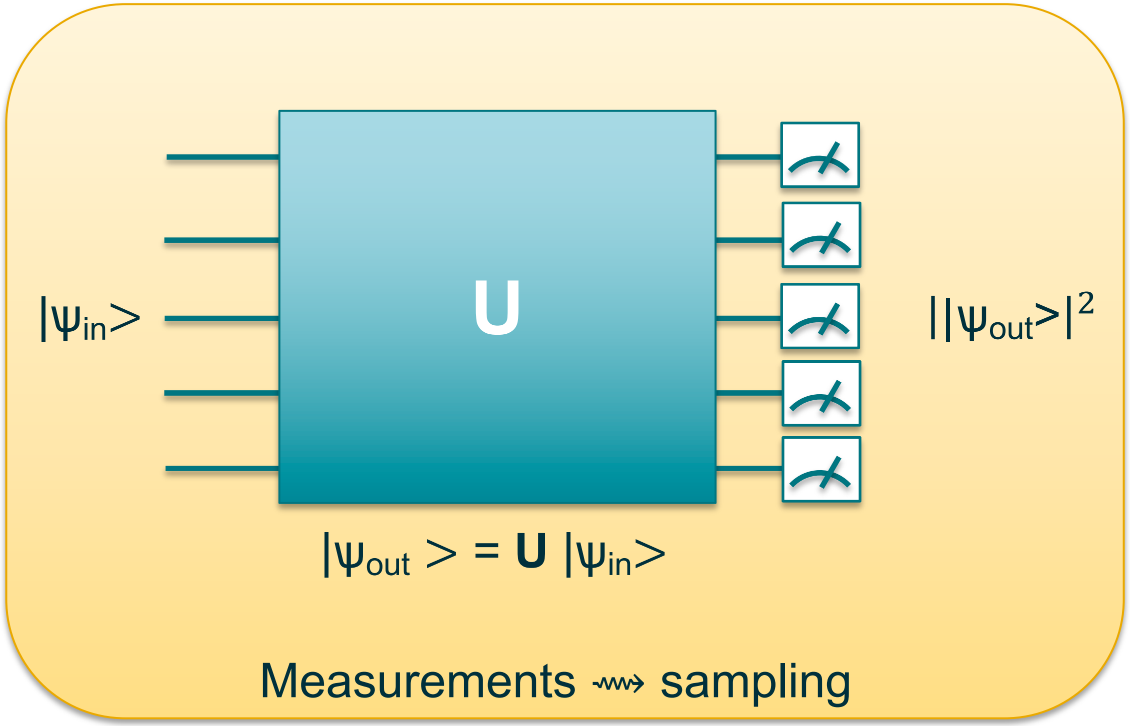
Conclusions

Problem statement

Quantum Circuits for Block-Encodings

Quantum computers perform unitary evolution

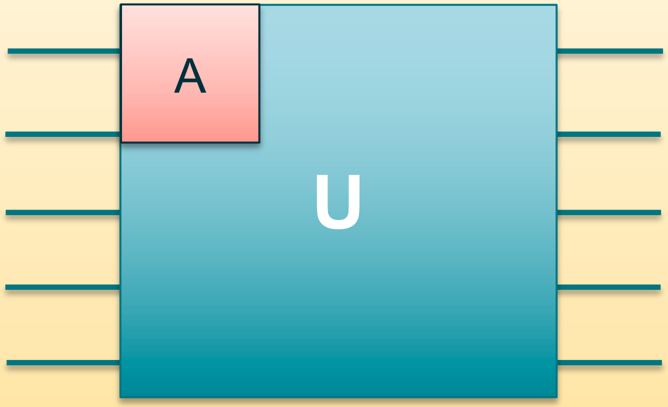




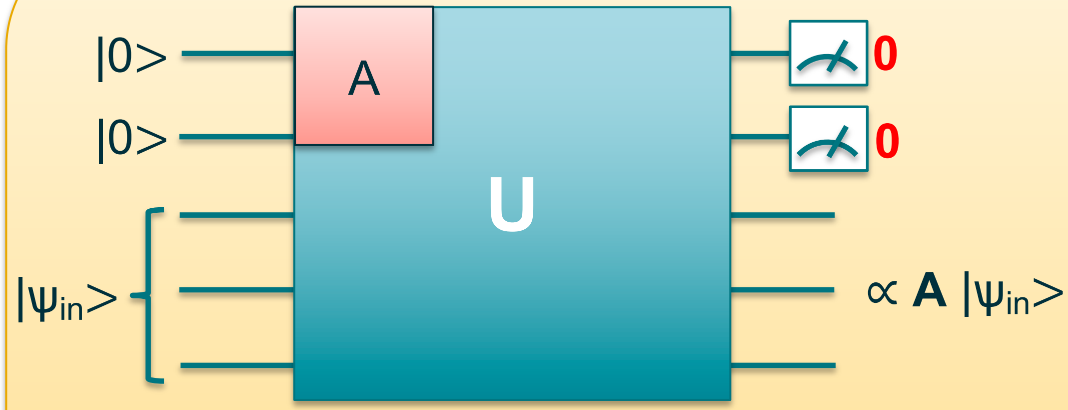
- Unitary evolution has to be **synthesized/compiled** to lower level **quantum circuit** description:
  - Generic (multi-)qubit gates > native gates > pulse sequence
- Many interesting problems are **not unitary** in nature

Block-encodings

A natural way to represent non-unitary transformation on quantum computers



Embed the non-unitary problem matrix **A** in the leading block of **U**



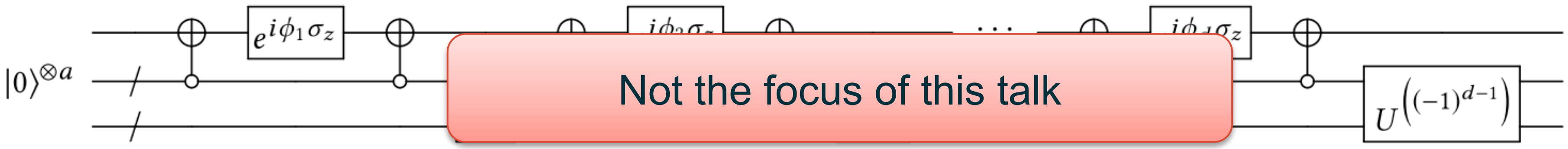
Prepares state proportional to **A** $|\psi_{in}\rangle$ if ancilla qubits are in zero state

- Wide range of applications: random walks, Hamiltonian simulation, quantum linear algebra, quantum machine learning, open quantum systems, thermal states, …

Background on block-encodings

A brief history of many (implicit) use cases

- **Implicitly** used by:
  - `Szegedy (2004)` in the context of quantized random walks
  - `Berry, Childs, Kothari (2015)` for Hamiltonian simulation
  - `Childs, Kothari, Somma (2017)` for quantum linear systems problem
  - …
- **Formalized** by `Gilyen, Su, Low, Wiebe (2018)` as part of seminal **quantum singular value transformation** algorithmic framework



$$U_{\widetilde{\Phi}} = e^{i\widetilde{\phi}_0 Z_\Pi} \prod_{j=1}^{d} (U_A e^{i\widetilde{\phi}_j Z_\Pi}) = \begin{pmatrix} P(A) & * \\ * & * \end{pmatrix}$$

`Low, Chuang (2019), Gilyen et al. (2019), Martin et al. (2021)`

Black box query access through oracles

A common assumption throughout the literature

- Matrix access through one or more black box quantum query oracles
- For example: **sparse matrices**

$$O_r : |i\rangle|k\rangle \rightarrow |i\rangle|r_{ik}\rangle$$

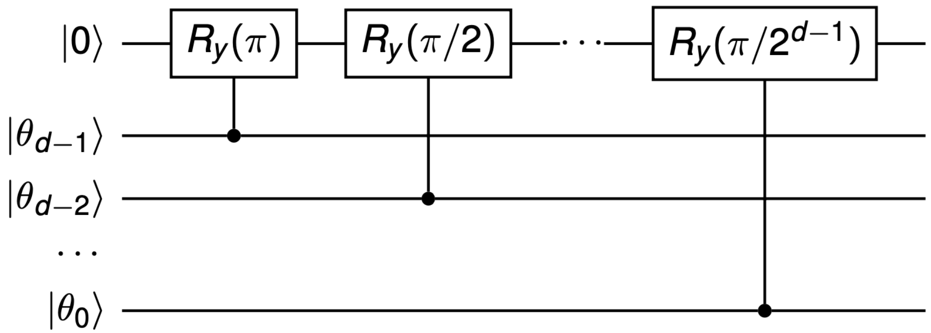
row index

$$O_c : |\ell\rangle|j\rangle \rightarrow |c_{\ell j}\rangle|j\rangle$$

column index

$$O_A : |i\rangle|j\rangle|0\rangle^{\otimes b} \rightarrow |i\rangle|j\rangle\boxed{|a_{ij}\rangle}$$

matrix values [binary bitstring encoding]



Not easy to come up with circuits for the query oracles

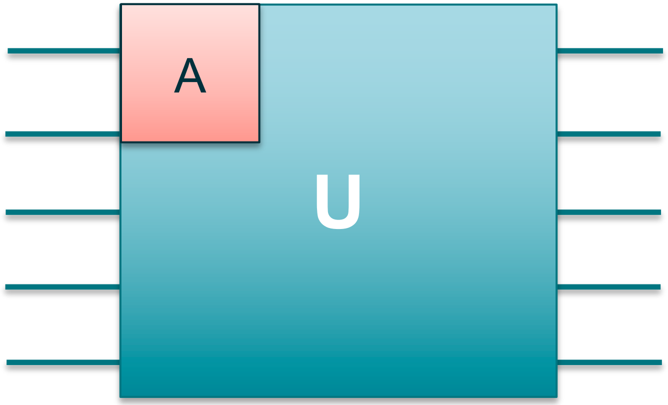
This talk

Direct encoding of the matrix data in a compact circuit

FABLE:

- takes in $N \times N$ matrix A
- generates a circuit that consists of only H, R_y, CX, SWAP gates
- classical complexity $O(N^2 \log N)$
- worst-case circuit gate complexity $O(N^2)$
  - – often significantly better for structured problems (see examples)

```
circuit = fable(A)
```



Uniformly controlled rotations

Key component for building matrix query oracles

Matrix query oracle

If we have access to an oracle that provides the matrix data in a superposition:

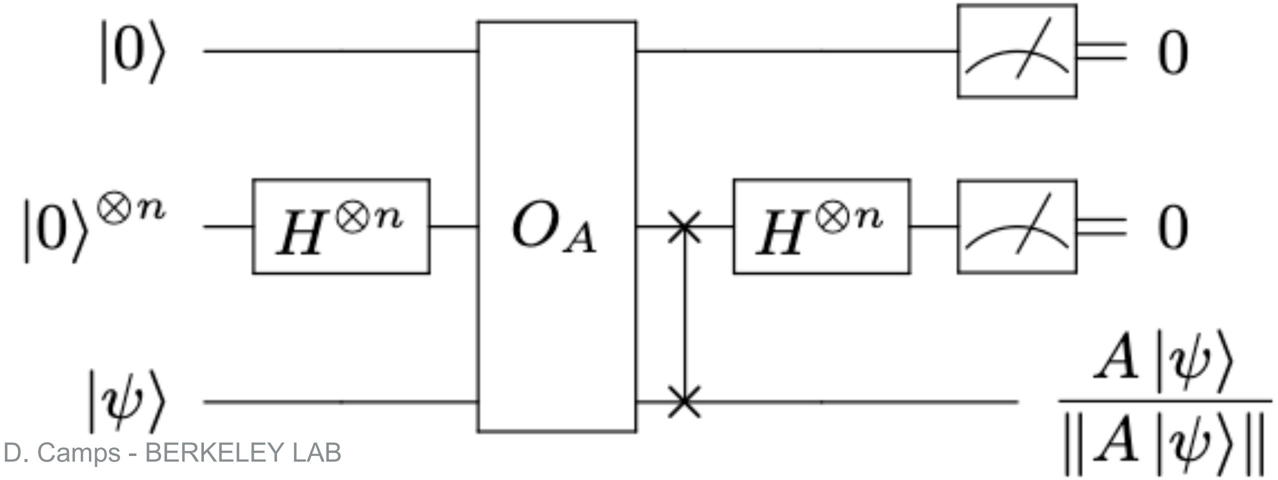
$$O_A \left|0\right\rangle \left|i\right\rangle \left|j\right\rangle = \left(a_{ij} \left|0\right\rangle + \sqrt{1 - |a_{ij}|^2} \left|1\right\rangle \right) \left|i\right\rangle \left|j\right\rangle$$

1 qubit

n qubits

n qubits

The following circuit block encodes the matrix:



Proof: see Theorem 1

How to implement the oracle?

For a real-valued 2×2 matrix

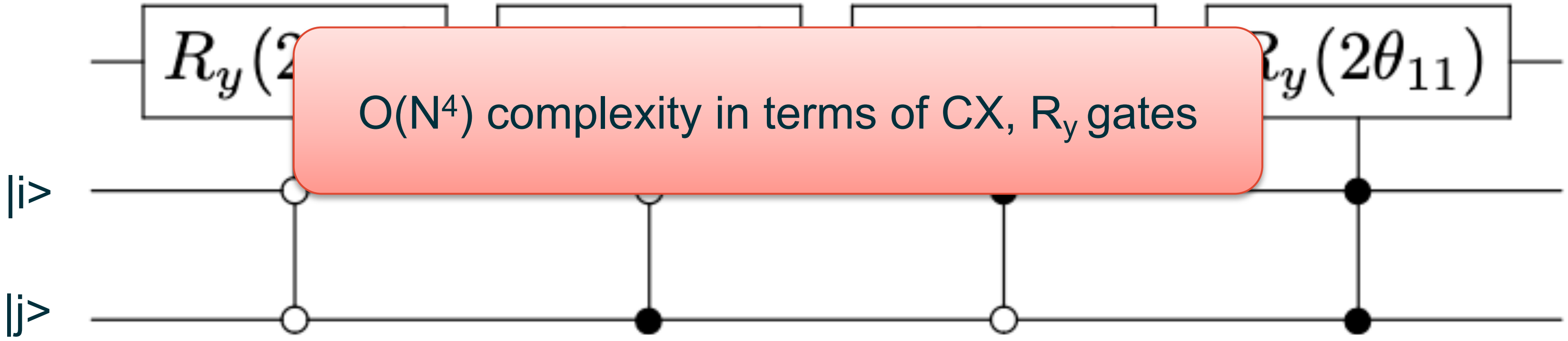
- Define the angles

$$\theta_{ij} = \arccos(a_{ij})$$

- R_y rotation

$$R_y(2\theta_{ij})\,|0\rangle = \begin{bmatrix} \cos(\theta_{ij}) & -\sin(\theta_{ij}) \\ \sin(\theta_{ij}) & \cos(\theta_{ij}) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a_{ij} \\ \sqrt{1 - a_{ij}^2} \end{bmatrix}$$

- Fully controlled rotation for every matrix element



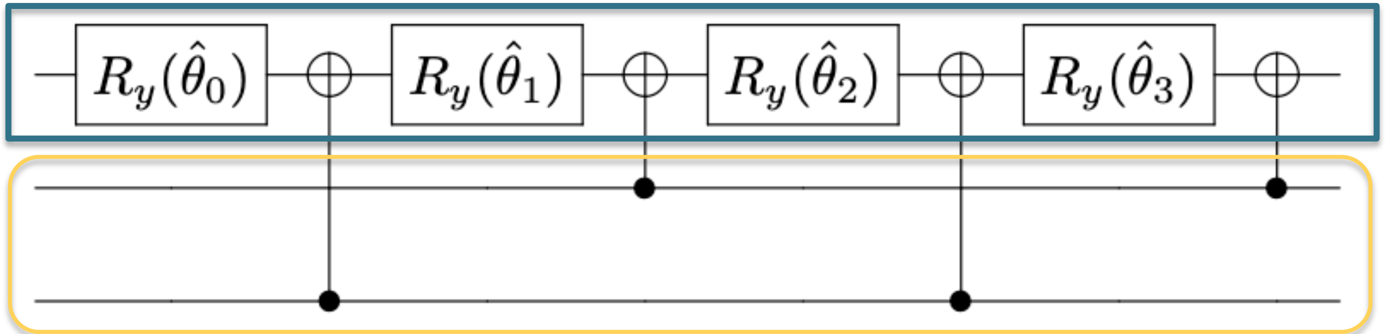
$$O_A\,|0\rangle\,|i\rangle\,|j\rangle = \left(a_{ij}\,|0\rangle + \sqrt{1 - |a_{ij}|^2}\,|1\rangle\right)|i\rangle\,|j\rangle$$

Uniformly controlled rotation

An efficient circuit implementation for the matrix oracle

- Introduced by **Möttönen et al. (2004)**
- Two key elementary properties of R_y rotations:

$$
\begin{aligned}
R_y(\theta_0)\, R_y(\theta_1) &= R_y(\theta_0 + \theta_1), \\
X\, R_y(\theta)\, X &= R_y(-\theta),
\end{aligned}
$$



$$
\begin{array}{ll}
00: & R_y(\hat{\theta}_3) \quad R_y(\hat{\theta}_2) \quad R_y(\hat{\theta}_1) \quad R_y(\hat{\theta}_0) = R_y(\hat{\theta}_3 + \hat{\theta}_2 + \hat{\theta}_1 + \hat{\theta}_0), \\
01: & R_y(\hat{\theta}_3) X R_y(\hat{\theta}_2) \quad R_y(\hat{\theta}_1) X R_y(\hat{\theta}_0) = R_y(\hat{\theta}_3 - \hat{\theta}_2 - \hat{\theta}_1 + \hat{\theta}_0), \\
10: & X R_y(\hat{\theta}_3) \quad R_y(\hat{\theta}_2) X R_y(\hat{\theta}_1) \quad R_y(\hat{\theta}_0) = R_y(-\hat{\theta}_3 - \hat{\theta}_2 + \hat{\theta}_1 + \hat{\theta}_0), \\
11: & X R_y(\hat{\theta}_3) X R_y(\hat{\theta}_2) X R_y(\hat{\theta}_1) X R_y(\hat{\theta}_0) = R_y(-\hat{\theta}_3 + \hat{\theta}_2 - \hat{\theta}_1 + \hat{\theta}_0),
\end{array}
$$

Computing the rotation angles

Structured linear system that can be solved efficiently

Linear system relates the angles in the circuit to the angles in the uniformly controlled rotation

$$\begin{bmatrix}\theta_0\\\theta_1\\\theta_2\\\theta_3\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 1\\1 & -1 & -1 & 1\\1 & 1 & -1 & -1\\1 & -1 & 1 & -1\end{bmatrix}\begin{bmatrix}\hat{\theta}_0\\\hat{\theta}_1\\\hat{\theta}_2\\\hat{\theta}_3\end{bmatrix}$$

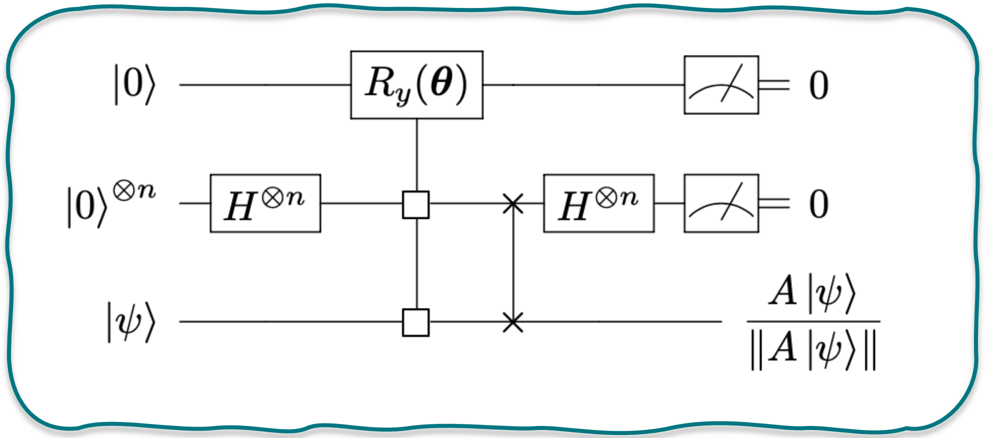
This can be rewritten as

$$\begin{bmatrix}\theta_0\\\theta_1\\\theta_2\\\theta_3\end{bmatrix} = \begin{bmatrix}1 & 1 & 1 & 1\\1 & -1 & 1 & -1\\1 & 1 & -1 & -1\\1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}1 & & & \\ & 1 & & \\ & & 0 & 1\\ & & 1 & 0\end{bmatrix}\begin{bmatrix}\hat{\theta}_0\\\hat{\theta}_1\\\hat{\theta}_2\\\hat{\theta}_3\end{bmatrix} = (\hat{H}\otimes\hat{H})P_G\begin{bmatrix}\hat{\theta}_0\\\hat{\theta}_1\\\hat{\theta}_2\\\hat{\theta}_3\end{bmatrix}$$

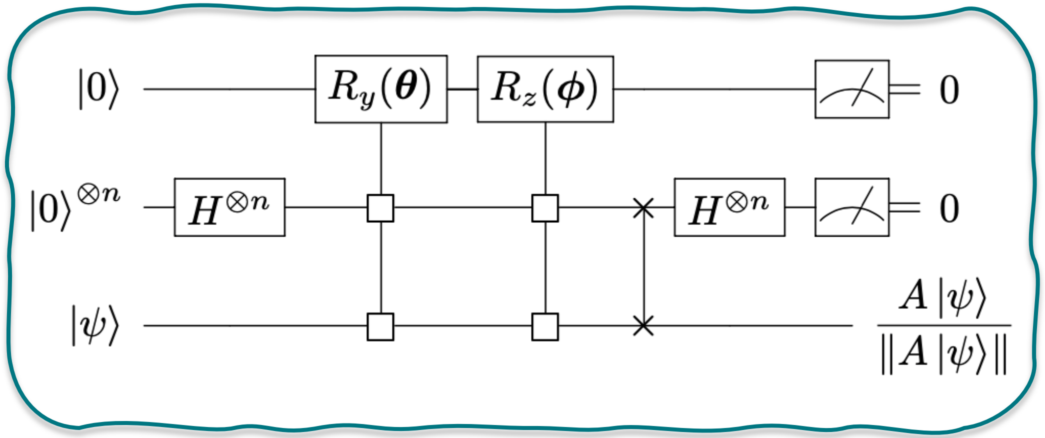
In general: $\left(\hat{H}^{\otimes 2n} P_G\right)\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}.$ → can be solved at O(N² log N)

Overall FABLE circuit

Uniformly controlled rotation circuit as matrix oracle



Real-valued matrix data



Complex-valued matrix data

- UCR_y : sets the magnitude of the matrix entries
- UCR_z : sets the phase of the matrix entries

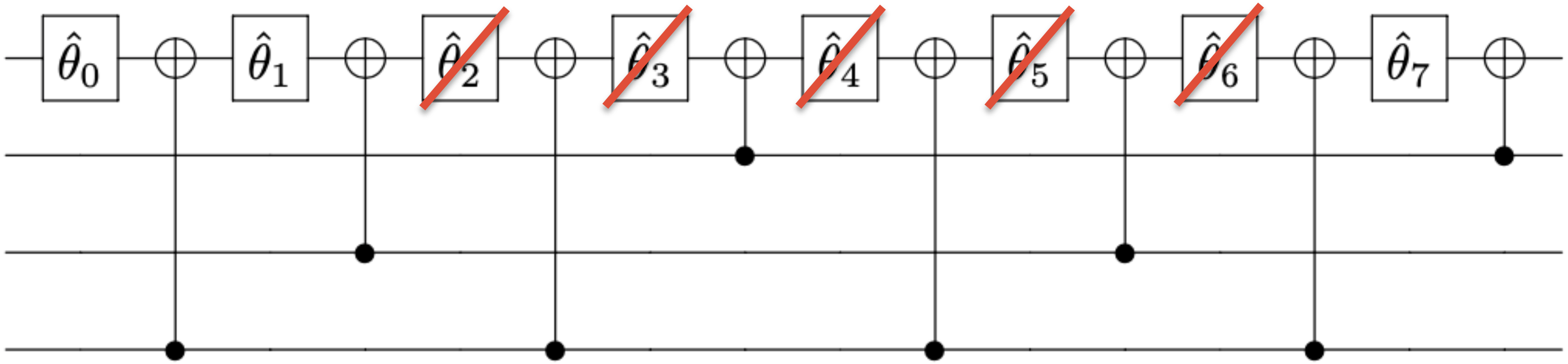
FABLE circuit for N × N matrix:

- $O(N^2 \log N)$ classical cost to compute angles
- $O(N^2)$ CX, Ry gates
- 2 log N + 1 qubits

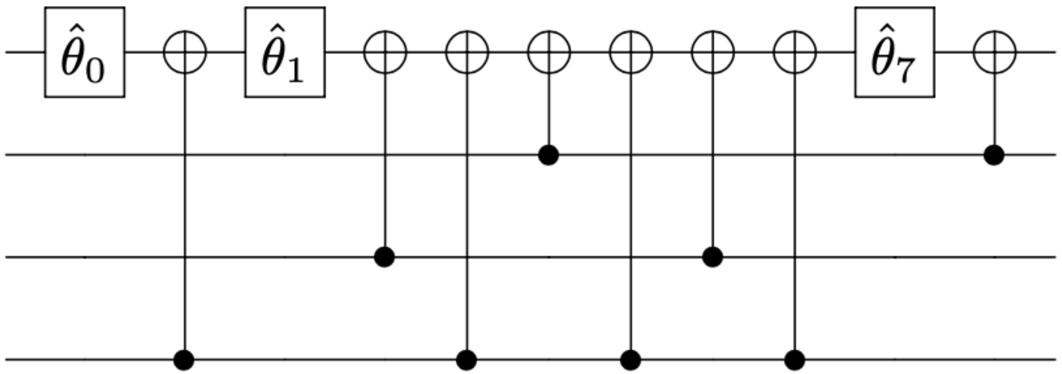
Compression theorem

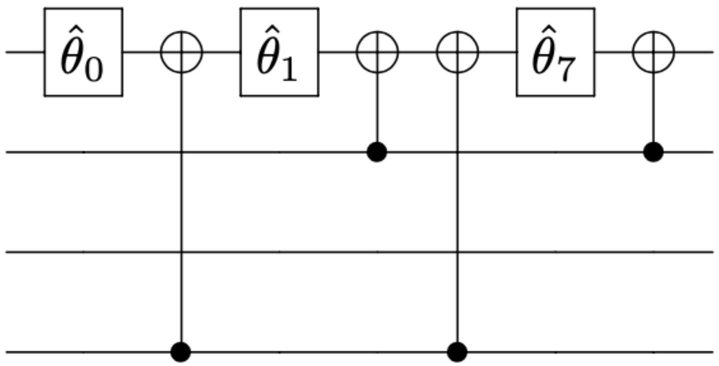
Circuit compression for approximate block-encodings

Compression of uniformly controlled rotations



$$\hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4, \hat{\theta}_5, \hat{\theta}_6 \leq \delta_c$$





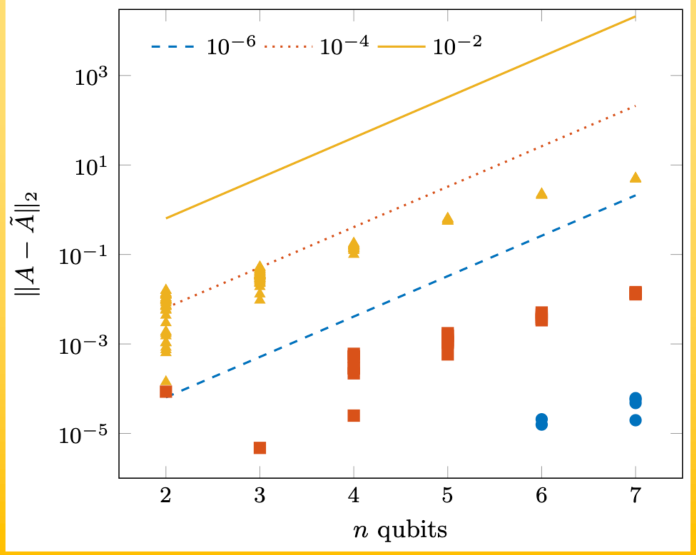
Compression theorem

Removing small angles leads to small perturbations on the block-encoding

- Compression threshold $\delta_c \geq 0$
- Input matrix $A \in \mathbb{R}^{N \times N}$

The error on the block-encoding is bounded from above by:

$$\left\|A - \tilde{A}\right\|_2 \leq N^3 \delta_c$$



- Pessimistic bound
- Many problems of interest have lots of angles smaller than δ_c

Examples

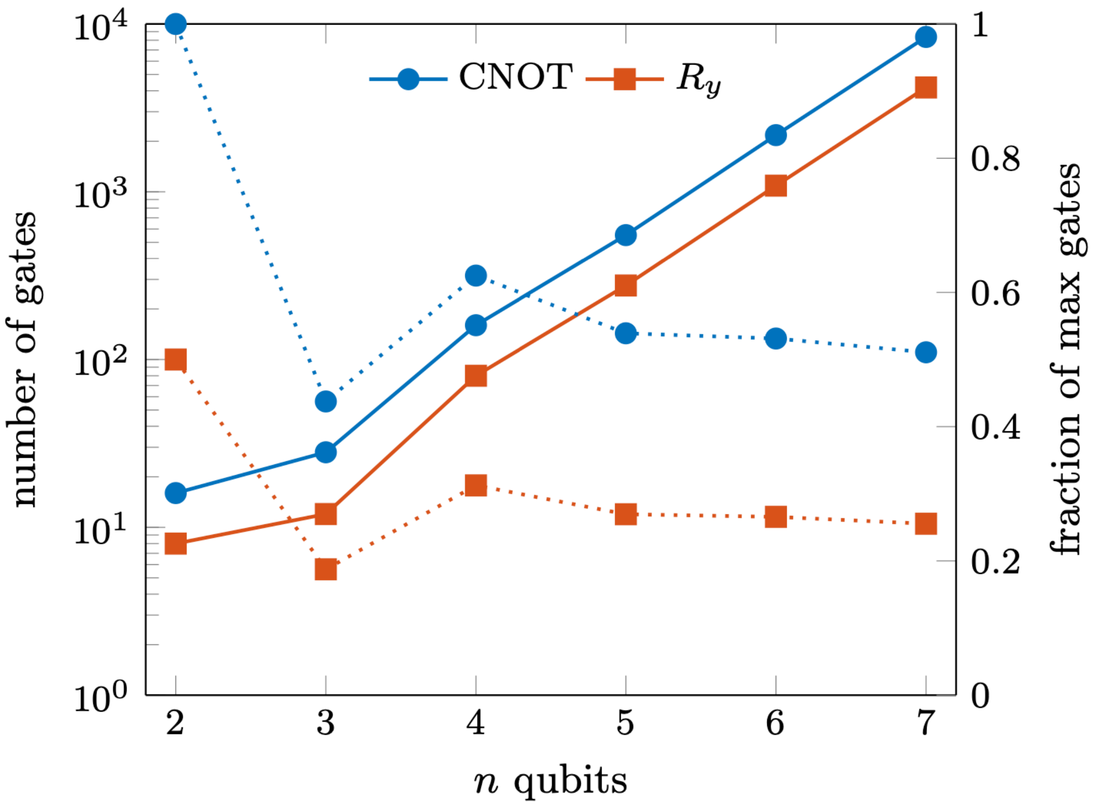
Localized Hamiltonians and PDEs

Heisenberg XXX model

Exact FABLE block-encoding $\delta_c = 0$

$$H = \sum_{i=1}^{n-1} J_x X^{(i)} X^{(i+1)} + J_y Y^{(i)} Y^{(i+1)} + J_z Z^{(i)} Z^{(i+1)}$$

$$J_x = J_y = J_z$$



Laplacian operators

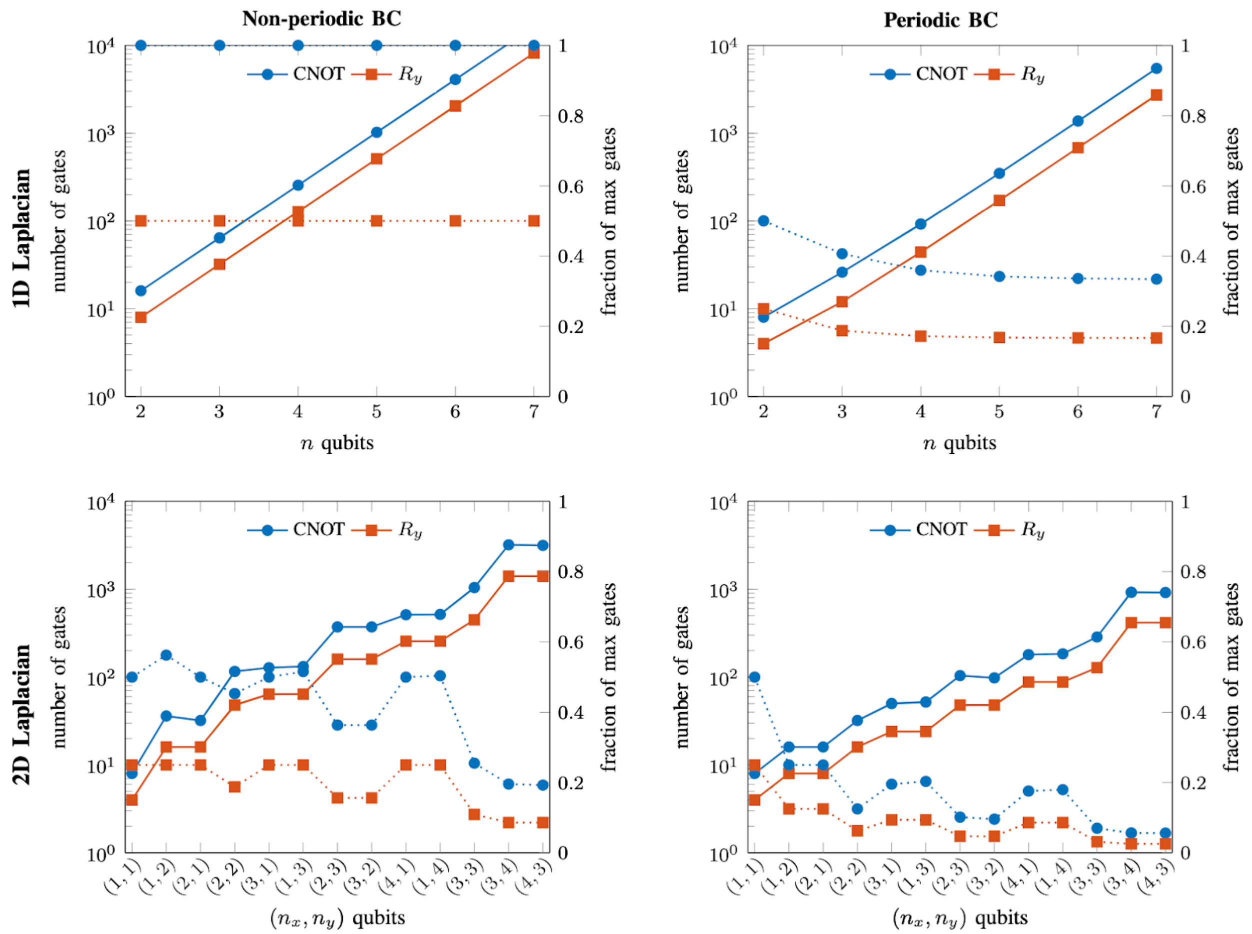
Exact FABLE block-encoding $\delta_c = 0$

1D:

$$L_{xx} = \begin{bmatrix} 2 & -1 & 0 & \cdots & * \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ * & \cdots & 0 & -1 & 2 \end{bmatrix}$$

2D:

$$L = L_{xx} \oplus L_{yy} = L_{xx} \otimes I + I \otimes L_{yy}$$



Application: Preparation of Canonical Thermal Pure Quantum States

Finite Temperature Properties on Quantum Computers

Preparation of canonical thermal pure quantum (TPQ) states

arXiv:2109.01619

Connor Powers

Lindsay Bassman Oftelie

Bert de Jong

Quantum system …

- … of size N
- … with Hamiltonian H
- … at inverse temperature β

The TPQ state is given by:

$$|\beta, N\rangle = \boxed{\hat{Q}}\,\boxed{|\Psi_R\rangle} \equiv \boxed{e^{-\beta H/2}}\,\boxed{|\Psi_R\rangle}$$

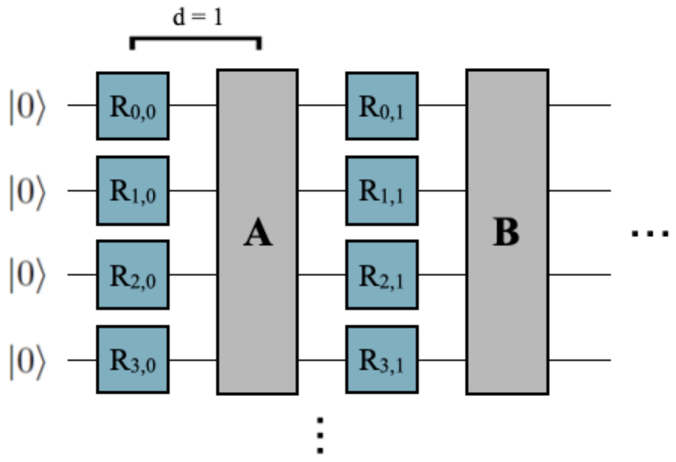
Haar-random pure state — **step 1**

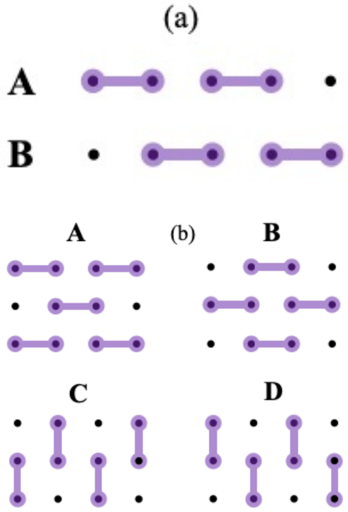
Non-unitary evolution — **step 2**

$$\langle \hat{A} \rangle_{\beta,N}^{ens} \equiv \frac{\mathrm{Tr}\,[e^{-\beta H}\hat{A}]}{\mathrm{Tr}\,[e^{-\beta H}]} \quad \leftrightarrow \quad \langle \hat{A} \rangle_{\beta,N}^{TPQ} \equiv \frac{\langle \beta, N|\,\hat{A}\,|\beta, N\rangle}{\langle \beta, N|\beta, N\rangle}$$

Two-step algorithm to prepare $|\beta, N\rangle$

1. Prepare Haar-random state





1D

2D

2. Non-unitary evolution
   – QITE / inexact QITE
   – Dilated unitary operator
   – FABLE

Comparison between circuit complexities

Tradeoff between gate complexity and number of ancilla qubits

CNOT Count

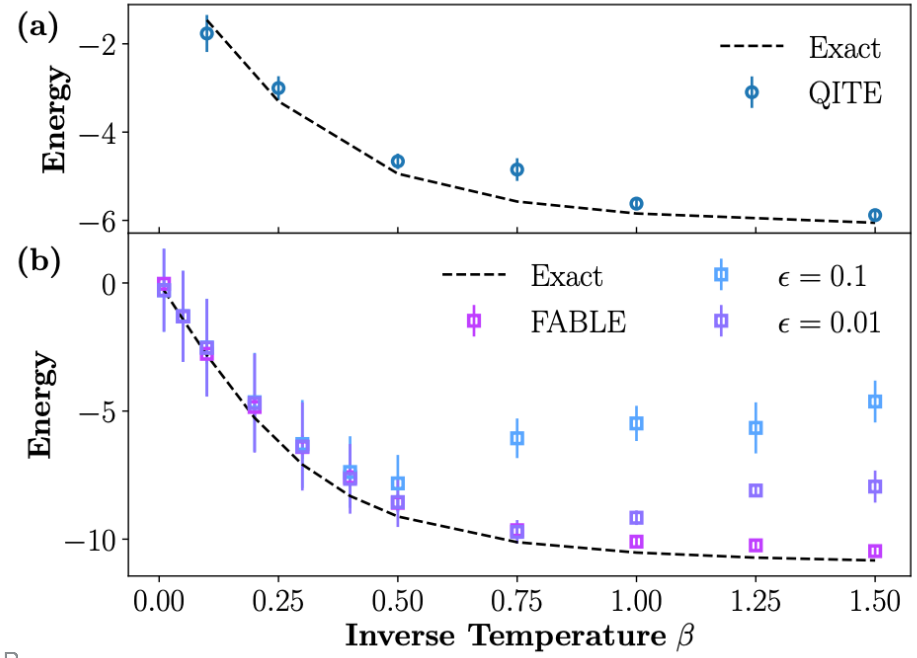
N	QITE	Inexact QITE	Dilated Operator	FABLE
2	14	20	41	16
3	97	963	218	64
4	-	1957	1025	256
5	-	2945	4474	1024

Ancillary Qubits

N	QITE	Inexact QITE	Dilated Operator	FABLE
2	0	0	1	3
3	0	0	1	4
4	0	0	1	5
5	0	0	1	6

Circuit Generation Time [s]

N	QITE	Inexact QITE	Dilated Operator	FABLE
2	2.85	0.719	0.970	$2.14\text{x}10^{-3}$
3	$1.44\text{x}10^{2}$	3.71	1.53	$6.39\text{x}10^{-3}$
4	-	7.53	4.44	$2.61\text{x}10^{-2}$
5	-	11.51	17.4	$8.71\text{x}10^{-2}$



Conclusions

A versatile tool for non-unitary evolution on quantum computers

- FABLE enables implementation of **non-unitary evolutions** on quantum hardware
- Closely related to "classical data loading problem"
- Quantum circuits **are fast and easy to generate up to ~15 qubits**
- Many problems have structure that FABLE circuits can exploit through **circuit compression**

Roel Van Beeumen

Want to try it out?

```
pip install fable-circuits
```

```
circuit = fable(A)
```

`https://github.com/QuantumComputingLab/fable`

Thank you for your attention!

Job alert

We're looking for 2 quantum algorithms postdocs to join our group:

- Quantum subspace methods
- Interdisciplinary team
- Theory, simulation, experimental

https://jobs.lbl.gov/jobs/quantum-algorithms-postdoctoral-fellow-5138

https://jobs.lbl.gov/jobs/quantum-algorithms-postdoctoral-fellow-5139

BERKELEY LAB

Tutorial alert

Berkeley Quantum Synthesis Toolkit

Friday at 10am (TUT24)

```
from bqskit import Circuit, compile
circuit = Circuit.from_file('in.qasm')
out_circuit = compile(circuit)
out_circuit.save('out.qasm')
```

BQSKit

bqskit.lbl.gov

Questions?