Optimizing quantum algorithms using matrix factorization

Marc Baboulin

MdS/LISN - Université Paris-Saclay

Collaboration with ATOS Quantum R&D Group

Saint-Girons, June 20, 2022

Outline

Quantum computing

Quantum computing = exploiting specific properties of quantum mechanics (superposition, entanglement) to perform computations.

- Some problems can be solved **faster** with a quantum computer: [Shor, 1994] , [Lloyd, 1996] , [Grover, 1998] .
- Some are beyond the reach of the most powerful existing supercomputers ("quantum supremacy").
- Targeted fields: quantum chemistry, biology, machine learning, cryptography, optimization...
- Exascale supercomputer probably beaten with only 60 (not noisy) qubits.
- Quantum technologies are much more energy-efficient: 21 MW (Frontier) vs 10^{-4} MW (Sycamore, 54 qubits).

Quantum promises

- **Principle:** a quantum device has a state (made up of qubits instead of bits) which is initialized and then evolves by applying given operations (algorithm).
  At the end, informations about the state are measured.
- **Parallelism:** operations on n qubits are computed on the whole superposition at the same time (2^n coeff. for n qubits) $\rightarrow$ potentially exponential speedup.
- **But:** return a single probabilistic result and the quantum technology is fragile: coherence time is very short, quantum computations are unreliable (NISQ).
- 2 computational models: gate/circuit model (in this talk) and quantum annealing model.
- Future computers will be hybrid (classical processor + specialized quantum device affected by noise).

Quantum vs classical computation

A quantum computer has a quantum register that contains qubits.

- bit $= \{0, 1\} \rightarrow$ qubit $\in \mathbb{C}^2$

  $|\psi\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha |0\rangle + \beta |1\rangle$, with $|\alpha|^2 + |\beta|^2 = 1$.

  $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ = state of the qubit = superposition of two basis states

  α, β = amplitudes

Quantum vs classical computation

A quantum computer has a quantum register that contains qubits.

- bit $= \{0, 1\} \rightarrow$ qubit $\in \mathbb{C}^2$

  $|\psi\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha |0\rangle + \beta |1\rangle$, with $|\alpha|^2 + |\beta|^2 = 1$.

  $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ = state of the qubit = superposition of two basis states

  α, β = amplitudes

- **n qubits** $\in \mathbb{C}^{2^n}$

  $|\psi\rangle \in \mathbb{C}^{2^n} = (\mathbb{C}^2)^{\otimes n} \Rightarrow$ dimension of state space grows exponentially with the number of qubits

  (in classical computing: state in $\{0, 1\}^n$).

  $|\psi\rangle = \sum_{s \in \{0,1\}^n} \alpha_s |s\rangle$,

  with $\alpha_s \in \mathbb{C}, s \in \{0, 1\}^n$ and $\sum_{s \in \{0,1\}^n} |\alpha_s|^2 = 1$.

  $|\psi\rangle$ will be the input of the quantum algorithm.

Operations on qubits

All computations are driven by linear algebra operations.

- qubits can be put side-by-side using tensor product:
$$\underbrace{|\psi_3\rangle}_{\mathbb{C}^{2^{n+m}}} = \underbrace{|\psi_1\rangle}_{\mathbb{C}^{2^n}} \otimes \underbrace{|\psi_2\rangle}_{\mathbb{C}^{2^m}}$$

- A state $|\psi\rangle \in \mathbb{C}^{2^n}$ that cannot be expressed as a product $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$ is said to be entangled.

- The evolution of a quantum state is governed by unitary transformations through matrix/vector products.
$$|\psi_{t_2}\rangle = U\,|\psi_{t_1}\rangle \iff |\psi_{t_1}\rangle = U^H\,|\psi_{t_2}\rangle$$
$$U \in \mathbb{C}^{2^n \times 2^n} \text{ is unitary } (U^H U = I).$$

$\Rightarrow$ quantum operations are linear, norm-preserving and reversible.

Gate-based quantum computer

- We are limited to gates/operators that are implementable, depend on the hardware.
- Examples of quantum operators on 1 qubit:

- $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ = "NOT" or "bit flip"
- $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
- $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$
- $R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$
- $R_x(\theta) = \begin{pmatrix} \cos\theta & -i\sin\theta \\ -i\sin\theta & \cos\theta \end{pmatrix}$

Gate-based quantum computer

- We are limited to gates/operators that are implementable, depend on the hardware.
- Examples of quantum operators on 1 qubit:

- $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ = "NOT" or "bit flip"
- $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
- $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$
- $R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$
- $R_x(\theta) = \begin{pmatrix} \cos\theta & -i\sin\theta \\ -i\sin\theta & \cos\theta \end{pmatrix}$

- Two-qubit gate: controlled-NOT (CNOT) $= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

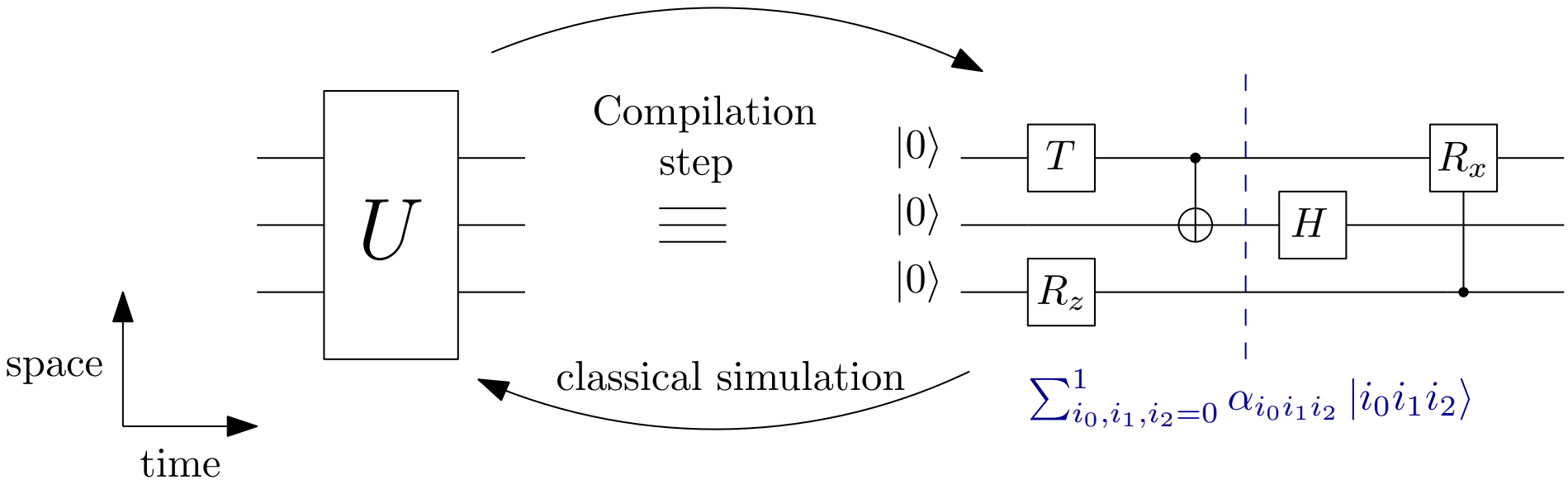
  Used to entangle 2 qubits:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \xrightarrow{CNOT} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \neq |\psi_1\rangle \otimes |\psi_2\rangle$$

Quantum circuit

Quantum algorithm = quantum circuit = series of quantum gates

Space composition $\rightarrow$ tensor product
Time composition $\rightarrow$ matrix multiplication from the left

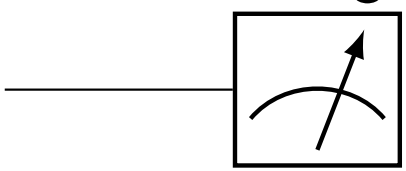


$$U = \Lambda_3(R_x) \times (I_2 \otimes H \otimes I_2) \times (CNOT \otimes I_2) \times (T \otimes I_2 \otimes R_z)$$

U is unknown to the hardware

Summary: quantum algorithm

- Initialize quantum state (vector of dimension 2^n).
- Execute the circuit (matrix of size $2^n \times 2^n$) $\equiv$ series of elementary actions applied on the quantum memory.
- No "quantum loop" or "conditional escape".
- No copying (no-cloning theorem).
- No direct access to the quantum state.
- Output state is measured (and then modified!) via a measurement gate (probabilistic):



- Post-process and interact with classical computer.

Outline

Quantum compilation

Traditionally, a compiler translates a user high-level language into instructions understandable by the machine.

In quantum computing this process is named circuit synthesis.

- **Linear algebra approach:**
  $U \in \mathcal{U}(2^n) \rightarrow U = \prod_i E_i$ (sequence of elementary matrices).
- The available elementary matrices are given by the hardware.
- The compilation is made using a classical computer (HPC).

Problem: Given a set of gates $\mathcal{S}$, find a circuit C such that:
- C contains only gates from $\mathcal{S}$,
- $\|U - U_C\| < \epsilon$ for some norm $\|\cdot\|$,
- with minimum computational time (= "classical" cost),
- with minimum number of gates (= "quantum" cost).

Quantum compilation: a hard problem

Without assumption on the matrix, both the number of operations (to decompose the matrix) and the number of resulting instructions (quantum gates) are exponential in n:

- $\mathcal{O}(4^n)$ for the number of instructions (gates),
- $\mathcal{O}(8^n)$ for the number of operations (flops).

$\Rightarrow$ the synthesis of a generic unitary matrix can be performed only on a limited number of qubits.

Application: encoding classical data in the quantum memory (quantity of data is exponential with the number of qubits).

Our method to decompose the quantum operator is based on QR decomposition of a unitary matrix.

QR factorization for unitary matrices

The complex QR of A leads to $A = QD$, with D diagonal.

$$A = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} \xrightarrow{H_1 A} \begin{pmatrix} e^{i\theta_1} & 0 & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} \xrightarrow{H_2 H_1 A} \begin{pmatrix} e^{i\theta_1} & 0 & 0 & 0 \\ 0 & e^{i\theta_2} & 0 & 0 \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{pmatrix}$$

$$\xrightarrow{H_3 H_2 H_1 A} \begin{pmatrix} e^{i\theta_1} & 0 & 0 & 0 \\ 0 & e^{i\theta_2} & 0 & 0 \\ 0 & 0 & e^{i\theta_3} & 0 \\ 0 & 0 & 0 & e^{i\theta_4} \end{pmatrix}$$

Two issues:

- choice of the Householder vector v_i ($H_i = I - \tau_i v_i v_i^H$):
  complex QR: $v_i = b \pm e^{i\theta}\|b\| e_1$ with b current column of A and $\theta = \arg(b_1)$.
  we choose: $v_i = b + e^{i\theta} e_1$ and $\tau_i = \frac{2}{\|v_i\|^2} = \frac{1}{1+|b_1|}$.
- update of the rest of the matrix

QR factorization - updating the matrix

$$A = \left(\begin{array}{c|c} a_{11} & r \\ \hline c & A'' \end{array} \right)$$

(v_1 : a_{11} and c; A' : r and A'')

Standard update rule :

$$HA' = A' - \frac{2}{\|v_1\|^2} v_1 \times (A'^H v_1)^H$$

New update rule (using orthonormality of the columns of A) :

$$HA'' = A'' - c \times r$$

Overall :

- the matrix/vector product is replaced by a rank-one update,
- twice less flops is needed : $\frac{2}{3} \times N^3 = \frac{2}{3} \times 8^n$ (QSD: 7×8^n),
- update optimized using BLAS 3 (ZTRMM and ZGEMM).

Producing a quantum circuit

Step I : QR factorization with Householder transformations

$$U = \prod_{i=1}^{2^n} H_i \times D \quad \text{with } D \text{ diagonal.}$$

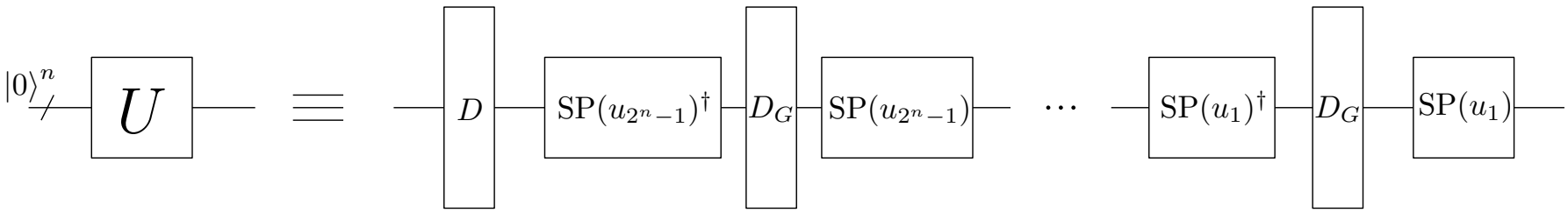
Step II : synthesis of Householder matrices

$$H_i = I - \frac{2}{\|v_i\|^2} v_i \times v_i^H = SP(v_i) \times D_G \times SP(v_i)^H$$

where :

- $SP(v_i) = \begin{pmatrix} v_i & | & * \end{pmatrix}$ prepares the Householder vector $v_i/\|v_i\|$,
- $D_G = \text{diag}(-1, 1, 1..., 1)$
  $(\text{because } H_i v_i = -v_i \text{ and } \forall u, u \perp v_i \Rightarrow H_i u = u)$.

Producing a quantum circuit

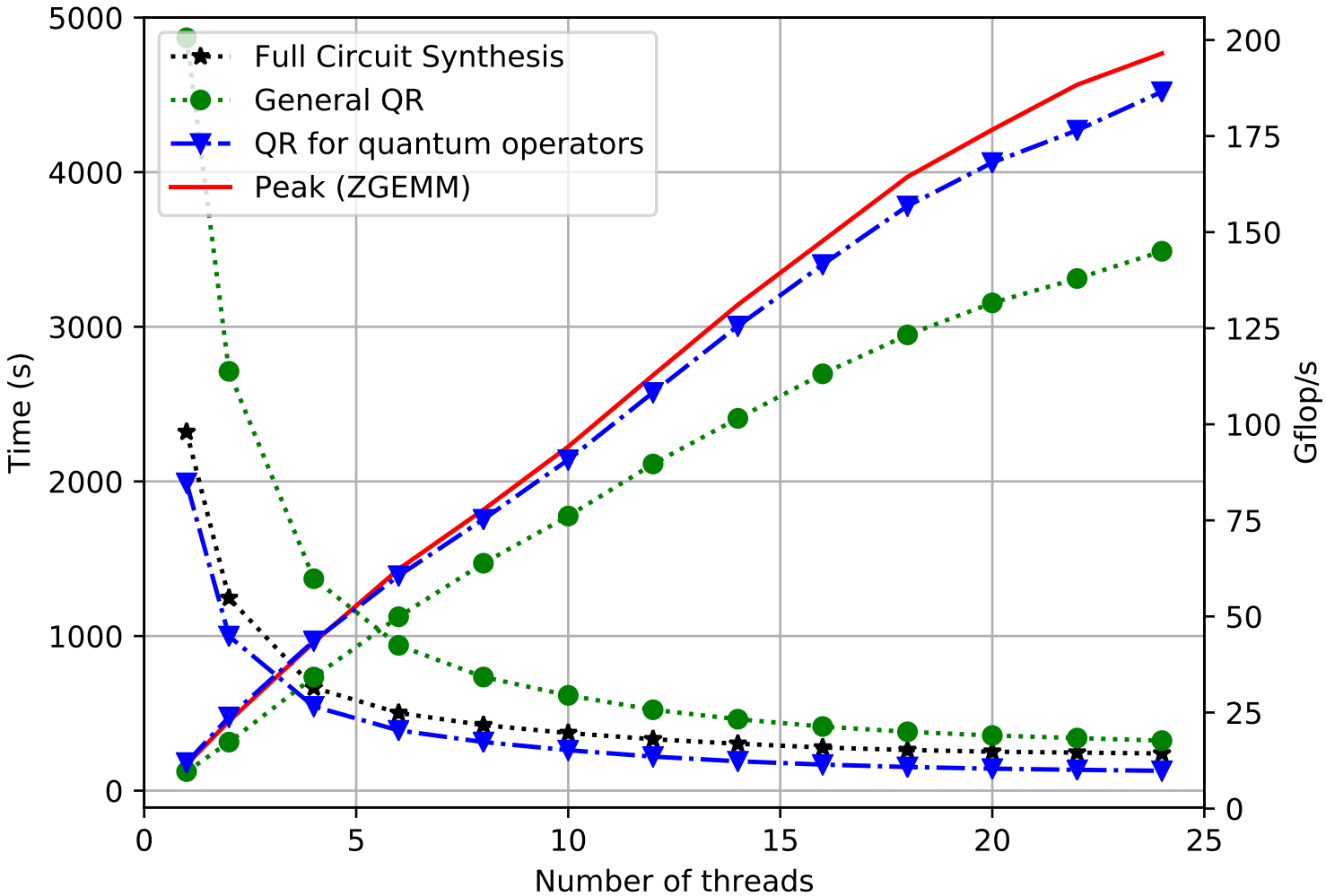


Step III : optimize adjacent state preparations/de-preparations

Gate and flop count :

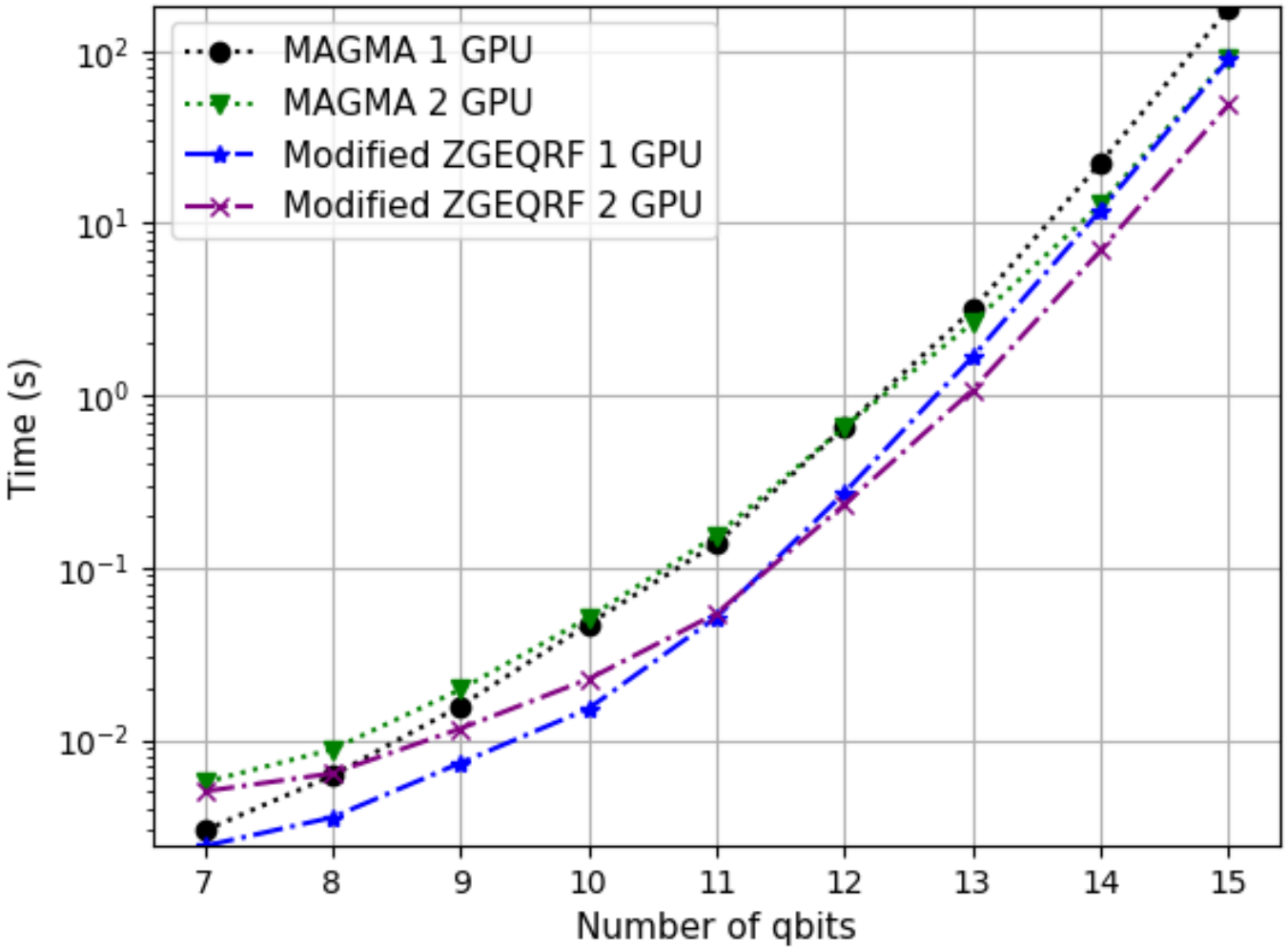
Computational cost $\simeq \frac{2}{3} \times 8^n$ (best, so far !).
Number of gates: 4^n (best is QSD method with $\sim \frac{4^n}{2}$).
Synthesis of D and D_G is negligible.

Experimental results - strong scaling



Strong scaling for operator decomposition and circuit synthesis on 15 qubits (24-core Intel Xeon(R) E7-8890 from Quantum Learning Machine at ATOS).

Experiments on GPUs



Performance on 12 Intel Xeon E5-2620 cores + 2 K40.

Outline

Conclusion

- Linear algebra is everywhere in quantum computing.
- Designing quantum (hybrid) algorithms requires a new way of thinking.
- HPC algorithms and platforms enable us to simulate and compile quantum circuits of intermediate size.
- Quantum computing requires minimizing classical resources (flops) in the compilation phase.
- We have decreased the computational cost for circuit synthesis thanks to an optimized QR factorization algorithm, [CPC, 2020] .
- Gate count can also be optimized using other type of factorization [TQC, 2021] , or BFGS [ICCS, 2019] .