Variational quantum architectures for linear algebra applications

Carlos Bravo-Prieto

IPAM Quantum Numerical Linear Algebra
27th January, 2022

Technology Innovation Institute
Universitat de Barcelona

Outline

- Quantum singular value decomposer: to produce singular value decomposition of bipartite pure states

- Variational quantum linear solver: for solving linear systems of equations

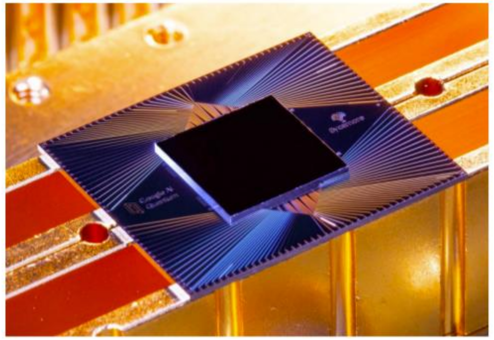
- Quantum generative models via adversarial learning: to learn underlying distribution functions.

Noisy intermediate-scale quantum (NISQ) era

NISQ era:

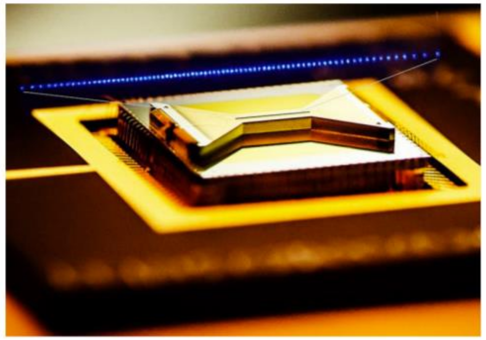
- Low number of qubits (50 qubits to a few hundreds)
- Low coherence times (~1000 operations)
- No error correction

Not yet capable of large-scale quantum computations



Google

IBM



IonQ

Variational quantum architectures

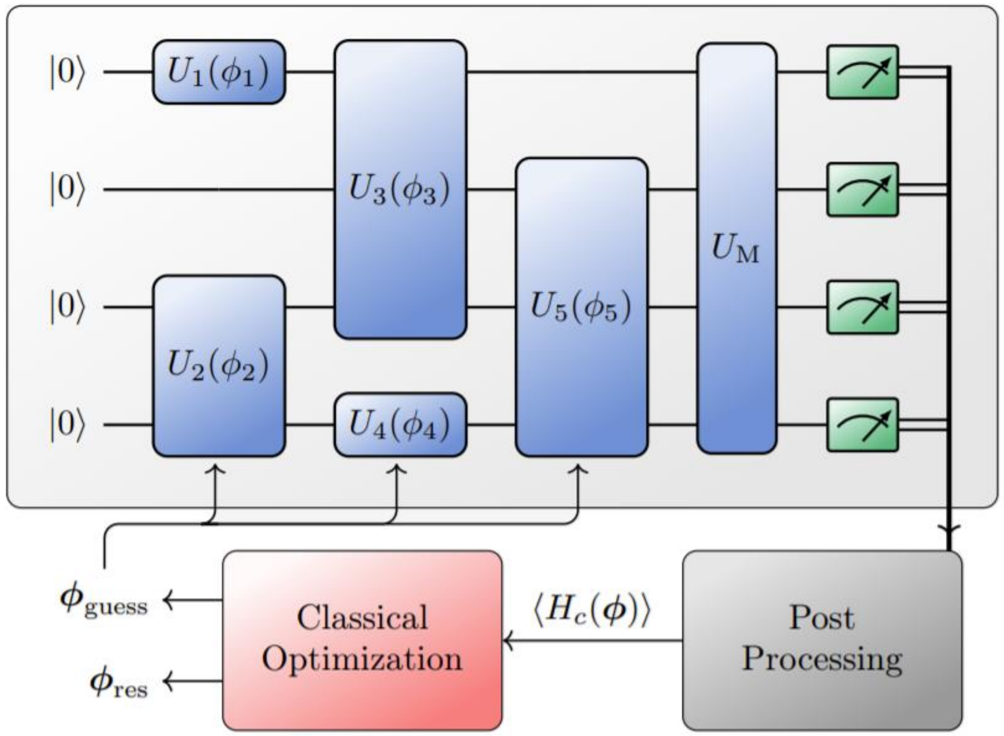
Candidates for near term advantage

- No high requisites in the number of qubits
- Shallow quantum circuits and hardware efficient
- Slightly noise resilience

Encode the problem into some cost function

Use a classical/quantum hybrid computation to minimize this cost function

$$\text{minimize}_{\phi} \; \langle \mathbf{0} | \, U(\phi)^{\dagger} H_c U(\phi) \, | \mathbf{0} \rangle$$



Quantum singular value decomposer

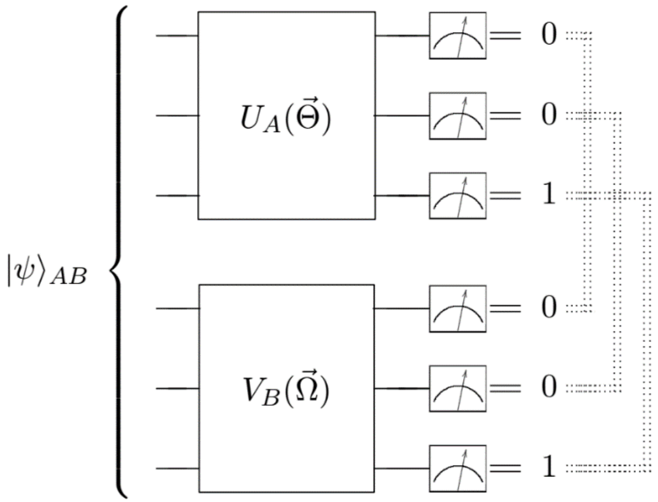
with D. García-Martín and J. I. Latorre, *Phys. Rev. A 101, 062310*

Quantum Singular Value Decomposer

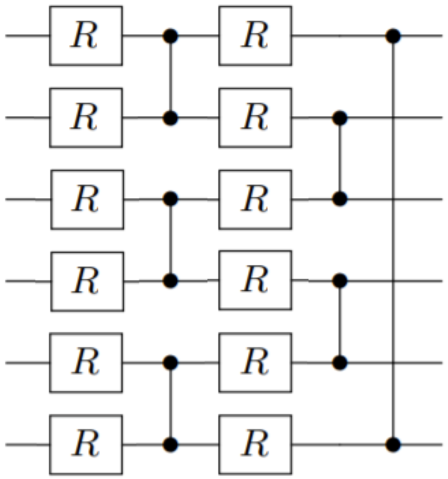
$$|\psi\rangle_{AB} = \sum_{i=1}^{\chi} \lambda_i \, |u_i\rangle_A |v_i\rangle_B$$

$$|\psi\rangle_{AB} \xrightarrow{QSVD} U_A(\vec{\Theta}) \otimes V_B(\vec{\Omega}) \, |\psi\rangle_{AB} = \sum_{i=1}^{\chi} \lambda_i \, e^{i\alpha_i} |e_i\rangle_A |e_i\rangle_B$$

Variational training to correlations



Only one measurement setting



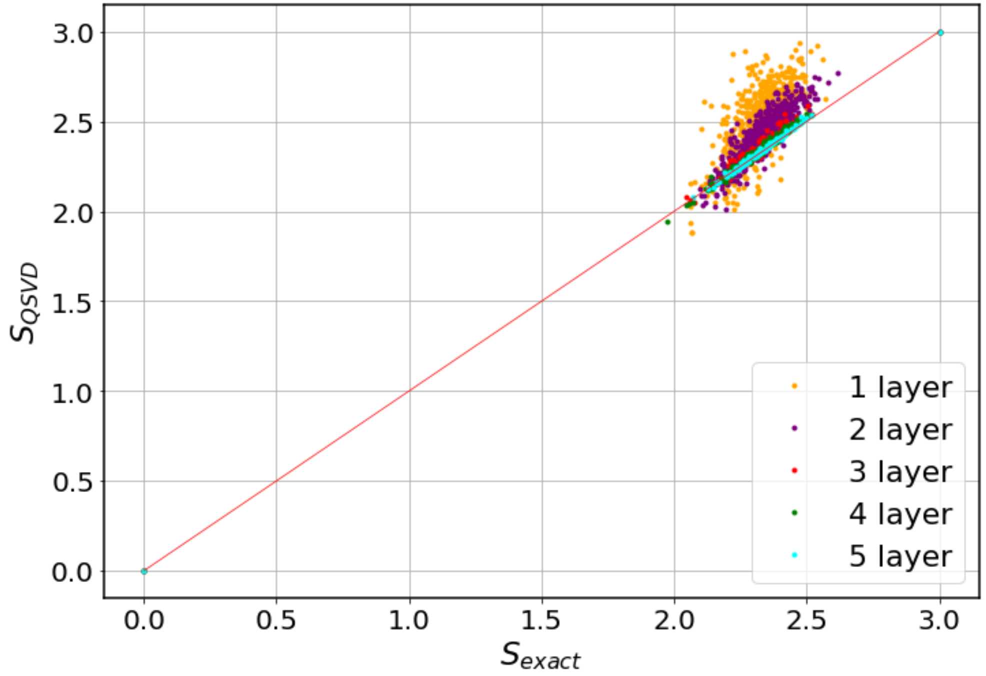
Once trained:

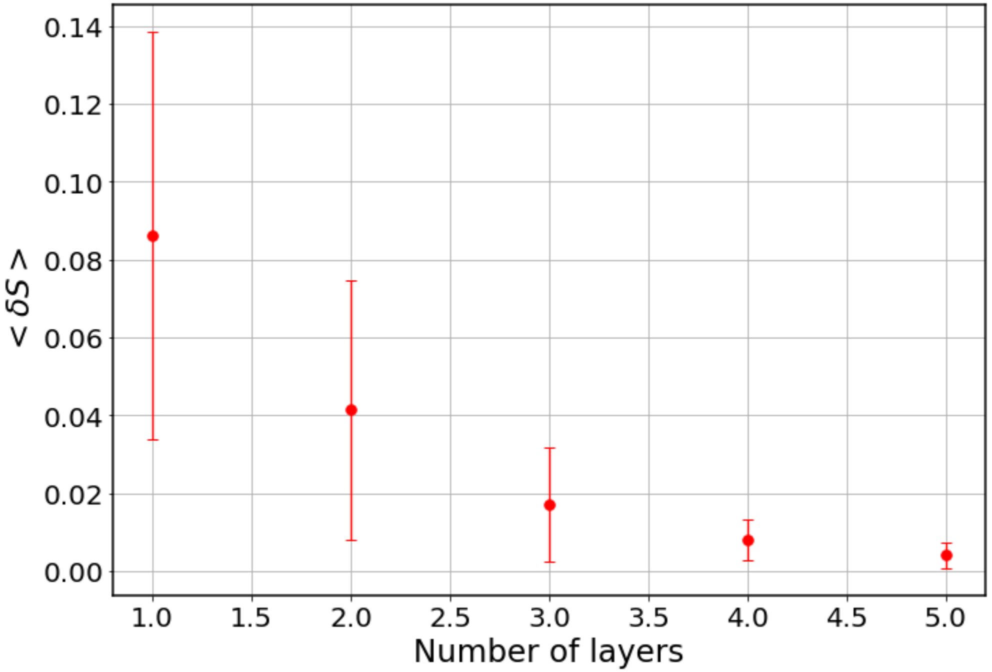
- Read out entropy spectrum

$$|\psi\rangle_{AB} = \sum_{i=1}^{\chi} \lambda_i \, |u_i\rangle_A |v_i\rangle_B$$

$$|\psi\rangle_{AB} \xrightarrow{QSVD} U_A(\vec{\Theta}) \otimes V_B(\vec{\Omega}) \, |\psi\rangle_{AB}$$

$$= \sum_{i=1}^{\chi} \lambda_i \, e^{i\alpha_i} |e_i\rangle_A |e_i\rangle_B$$





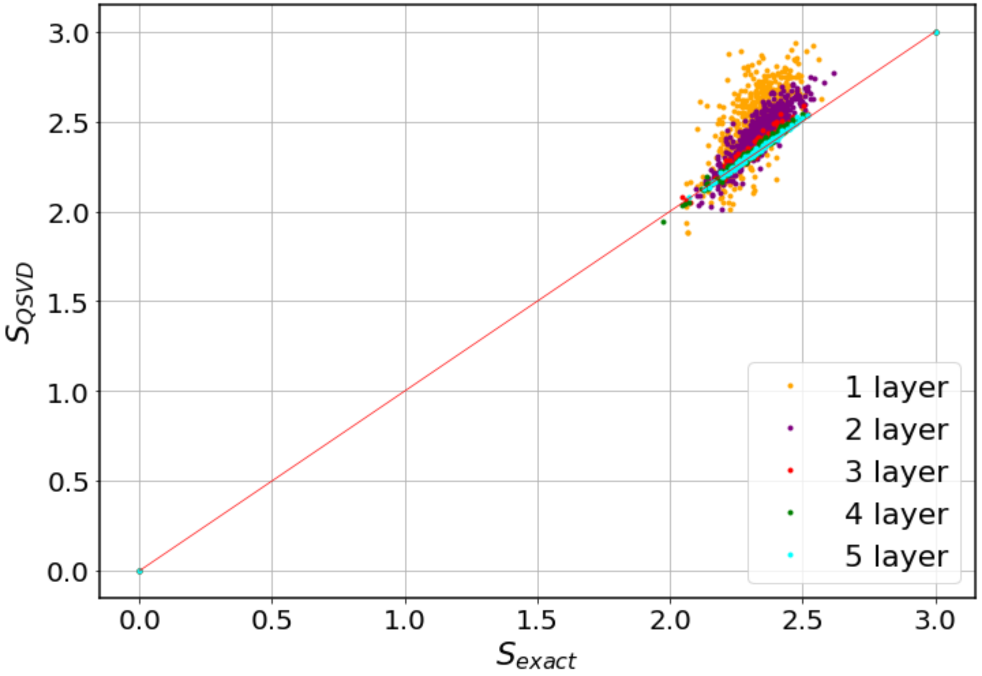
Once trained:

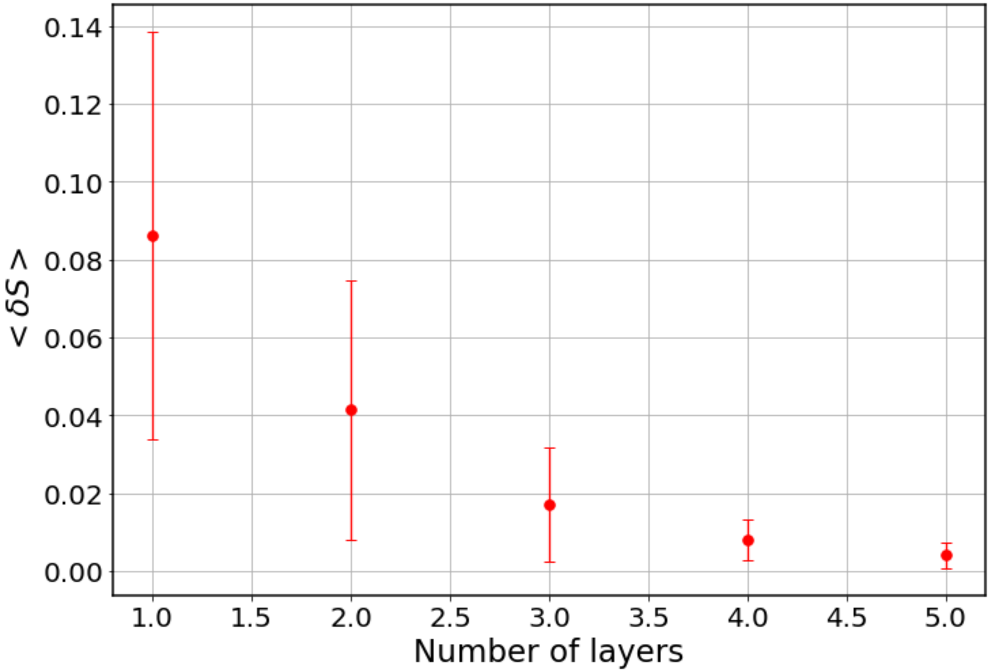
- Read out entropy spectrum
- Recover eigenvectors with inverted unitaries

$$|\psi\rangle_{AB} = \sum_{i=1}^{\chi} \lambda_i \, |u_i\rangle_A |v_i\rangle_B$$

$$|\psi\rangle_{AB} \xrightarrow{QSVD} U_A(\vec{\Theta}) \otimes V_B(\vec{\Omega}) \, |\psi\rangle_{AB}$$

$$= \sum_{i=1}^{\chi} \lambda_i \, e^{i\alpha_i} |e_i\rangle_A |e_i\rangle_B$$



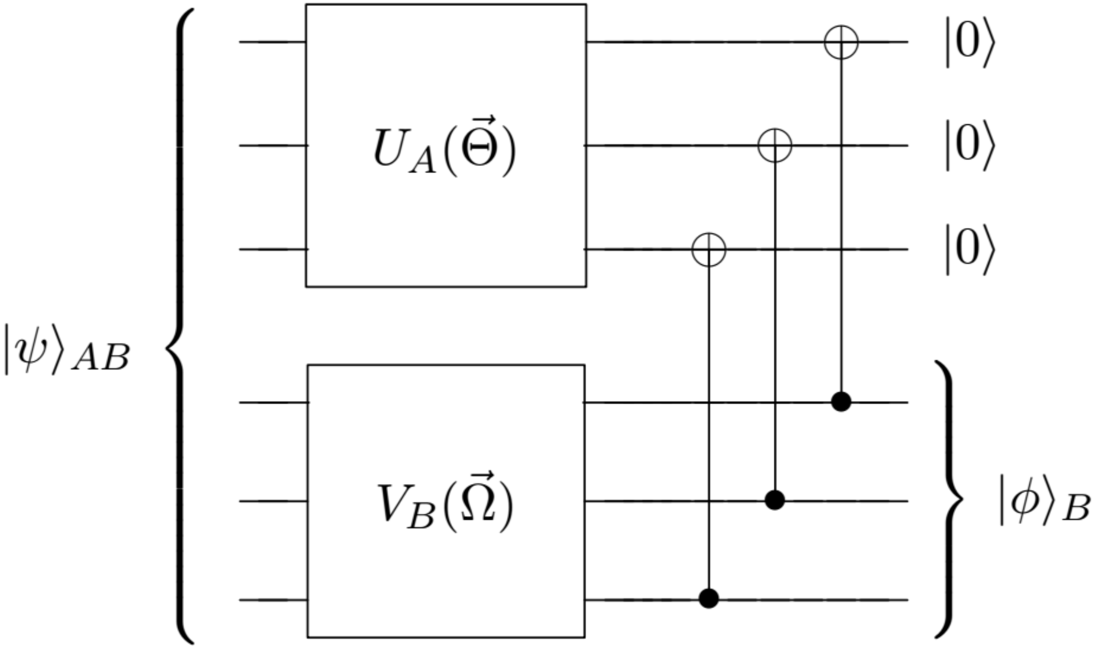


Once trained:
- Read out entropy spectrum
- Recover eigenvectors with inverted unitaries
- Autoencoder and SWAP

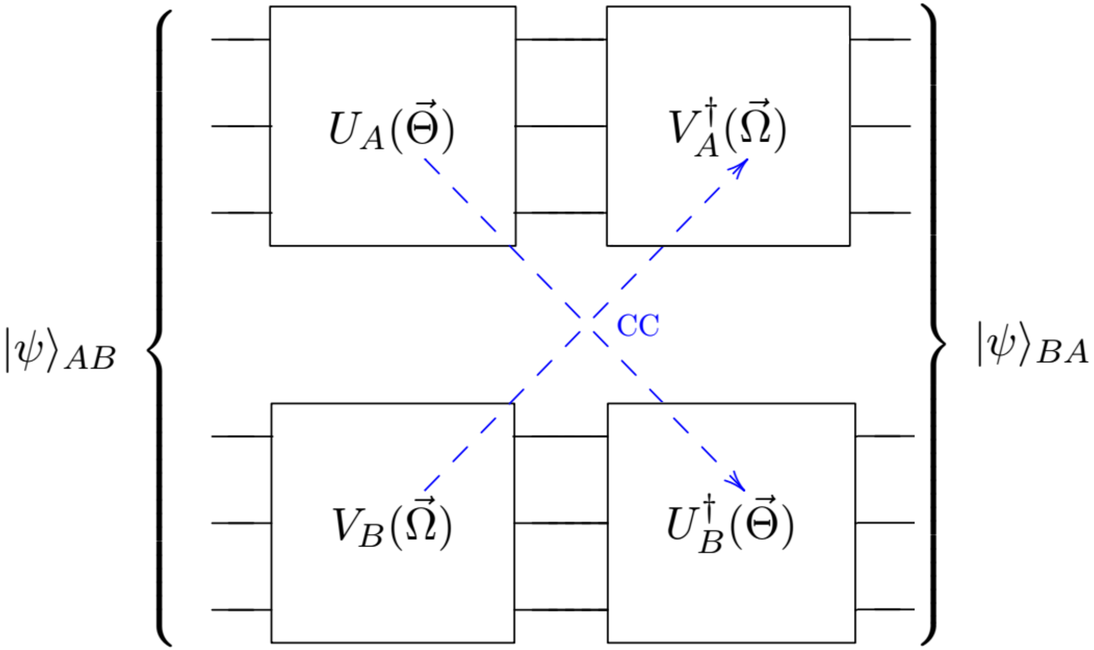
$$|\psi\rangle_{AB} = \sum_{i=1}^{\chi} \lambda_i \, |u_i\rangle_A |v_i\rangle_B$$

$$|\psi\rangle_{AB} \xrightarrow{QSVD} U_A(\vec{\Theta}) \otimes V_B(\vec{\Omega}) \, |\psi\rangle_{AB} = \sum_{i=1}^{\chi} \lambda_i \, e^{i\alpha_i} |e_i\rangle_A |e_i\rangle_B$$

Autoencoder



Long-distance SWAP



Variational quantum linear solver

with R. LaRose, M. Cerezo, Y. Subasi, L. Cincio and P. J. Coles, arXiv:1909.05820

Variational Quantum Linear Solver

$\boldsymbol{Ax} = \boldsymbol{b}$, where *A* is an *NxN* matrix

- Machine learning
- Partial differential equations
- Polynomial curve fitting
- Analyzing electrical circuits
- …

Classical algorithms: polynomial scaling in *N*

C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, arXiv:1909.05820

Variational Quantum Linear Solver

$\boldsymbol{Ax} = \boldsymbol{b}$, where *A* is an *NxN* matrix

Quantum algorithm: Harrow-Hassidim-Lloyd (HHL)

- Prepare |x>, such that |x> ~ ***x***
- Log *N* scaling
- Further improvements: reduced complexity in κ and ε
- Requires deep circuits

Variational quantum linear solver: geared towards NISQ

C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, arXiv:1909.05820

Variational Quantum Linear Solver

- **Define cost function**

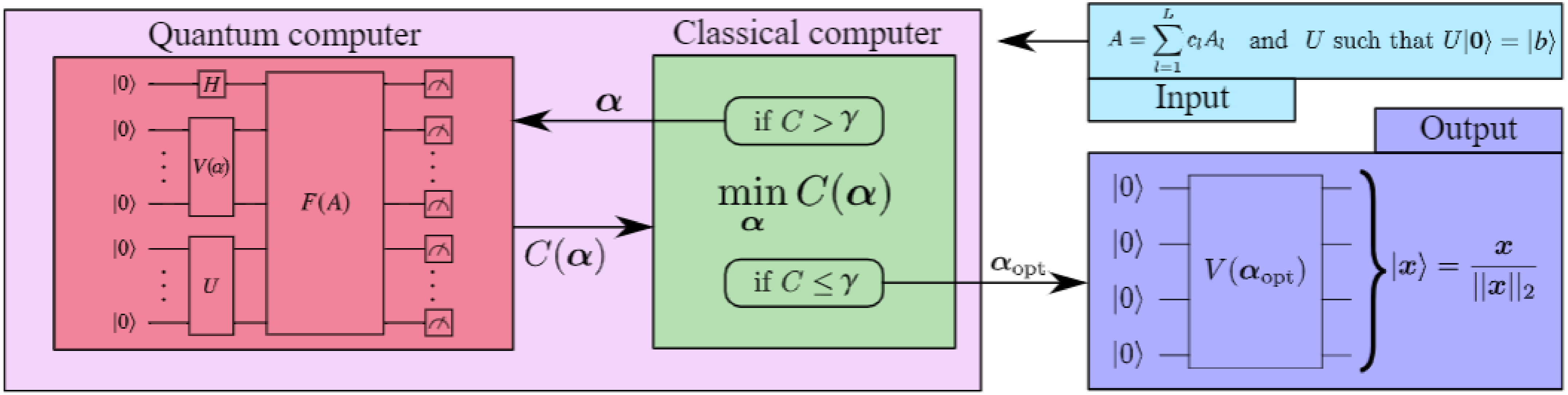
  C = 0 ⟶ *You solved the linear system!*

- **Operational meaning of C (e.g. solution guarantees)**

- **Find a circuit that computes C**
  - Efficient quantumly
  - Hard classically

C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, arXiv:1909.05820

Variational Quantum Linear Solver



C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, arXiv:1909.05820

VQLS: input

$$A = \sum_{l=1}^{L} c_l A_l \quad \text{and} \quad U \text{ such that } U|\mathbf{0}\rangle = |b\rangle$$

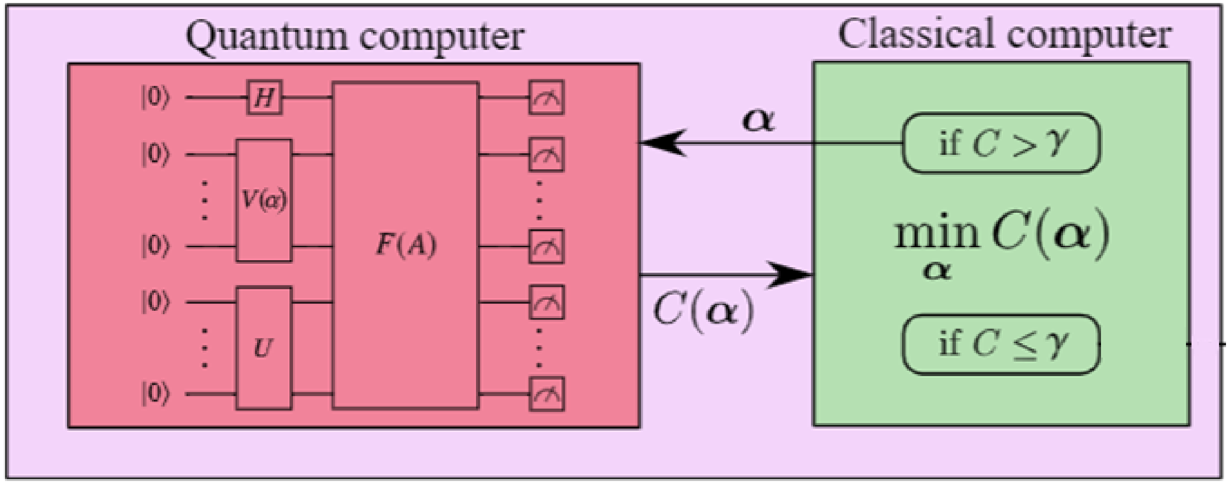
Input

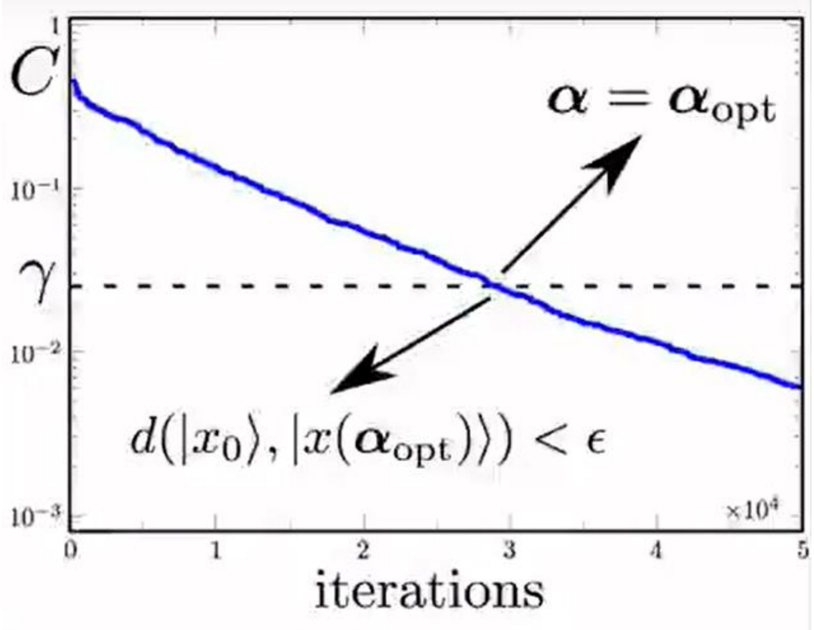
- Specify linear problem: $A|x\rangle \propto |b\rangle$
- Efficient circuit U: $U|0\rangle = |b\rangle$
- *A* is given by a linear combination of unitaries

$$A = \sum_{l} c_l A_l \ , \quad ||A|| \leq 1 \ , \kappa < \infty$$

C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, arXiv:1909.05820

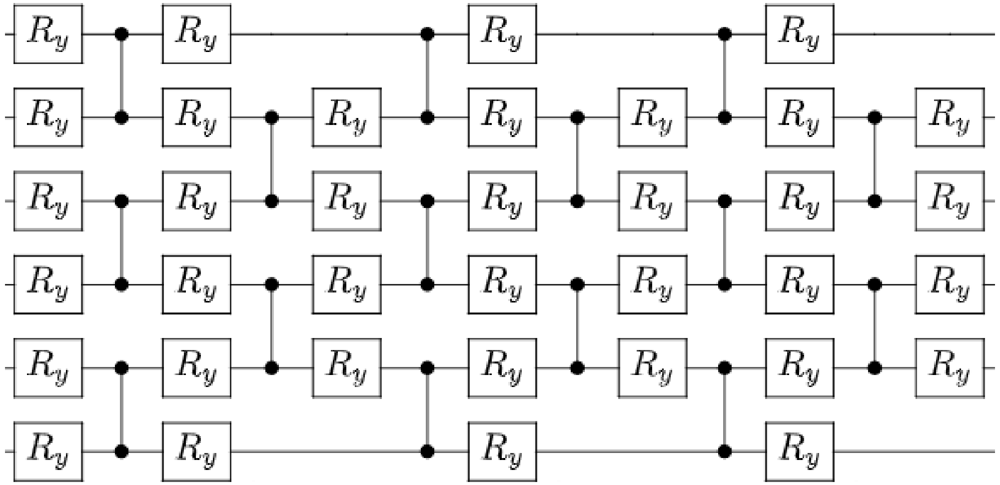
VQLS: optimization





- Goal: prepare $|x\rangle$ such that $A|x\rangle \propto |b\rangle$

- Ansatz for $|x\rangle$: $|x(\boldsymbol{\alpha})\rangle = V(\boldsymbol{\alpha})|\mathbf{0}\rangle$

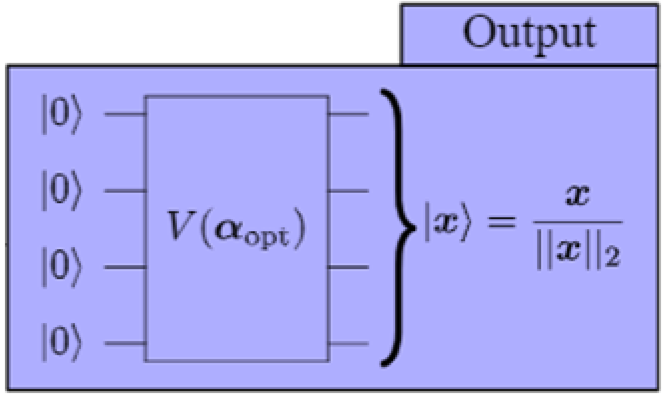


C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, arXiv:1909.05820

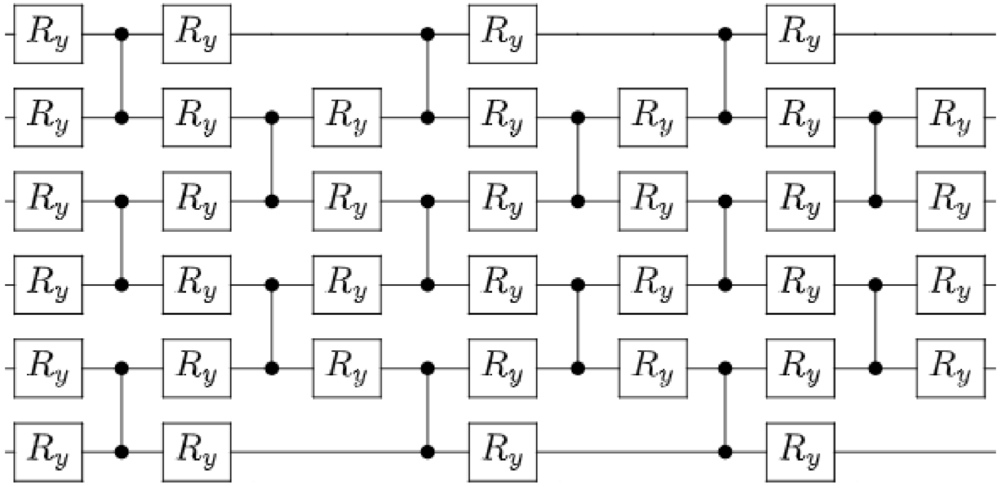
VQLS: output

- Optimal parameters $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{\text{opt}}$

- Prepare $|x(\boldsymbol{\alpha}_{\text{opt}})\rangle = V(\boldsymbol{\alpha}_{\text{opt}})|\mathbf{0}\rangle$



$V(\boldsymbol{\alpha}_{\text{opt}}) =$



C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, arXiv:1909.05820

VQLS: Cost functions

- Global cost function

$$C_G = \langle x | H_G | x \rangle$$

$$H_G = A^\dagger (\mathbb{1} - |b\rangle\langle b|) A$$

- Local cost function

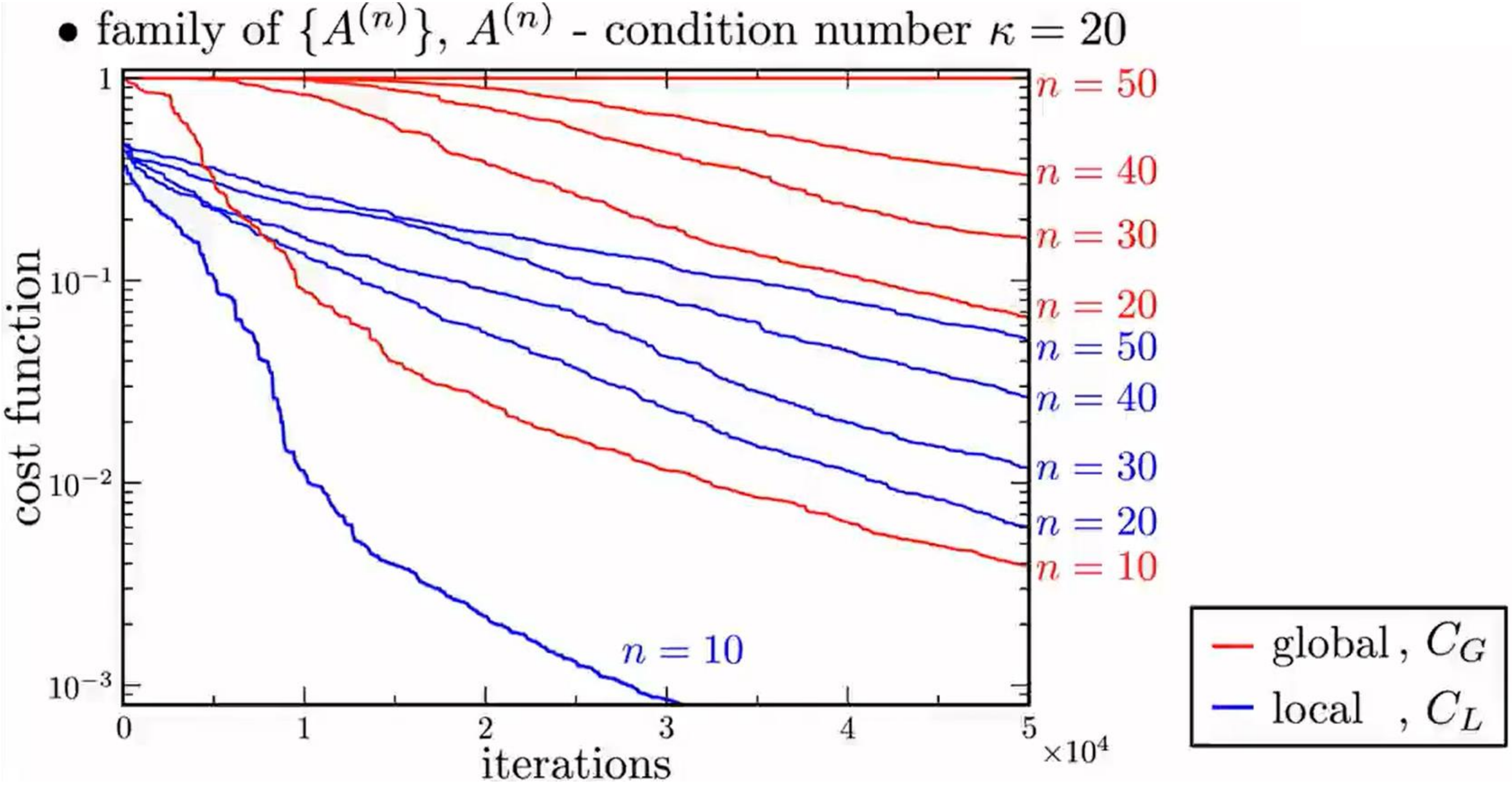
$$C_L = \langle x | H_L | x \rangle$$

$$H_L = A^\dagger U \left(\mathbb{1} - \frac{1}{n} \sum_{j=1}^{n} |0_j\rangle\langle 0_j| \otimes \mathbb{1}_{\bar{j}} \right) U^\dagger A$$

- $C_L \leqslant C_G \leqslant n C_L$

$$C_L = 0 \Longleftrightarrow C_G = 0 \Longleftrightarrow A|x\rangle \sim |b\rangle$$

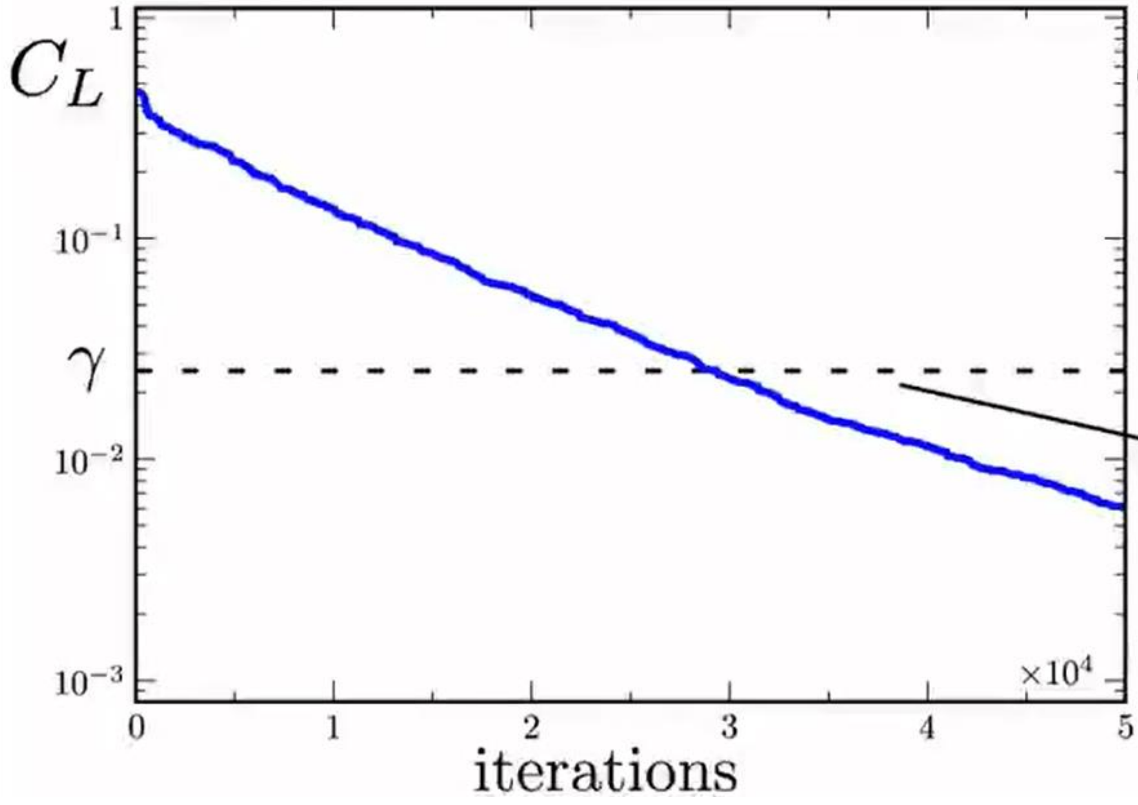
Barren plateaus: global vs local



Operational meaning

- one can show that

$$C_G \geq \frac{\epsilon^2}{\kappa^2} \qquad \text{and} \qquad C_L \geq \frac{1}{n}\frac{\epsilon^2}{\kappa^2}$$



$$\epsilon = \frac{1}{2}\mathrm{Tr}\left|\,|x_0\rangle\langle x_0| - |x(\boldsymbol{\alpha}_{\mathrm{opt}})\rangle\langle x(\boldsymbol{\alpha}_{\mathrm{opt}})|\,\right|$$

$|x_0\rangle$ - exact solution

$$C_L = \gamma$$

$$\epsilon \leq \kappa\sqrt{n\gamma}$$

Example: simulations

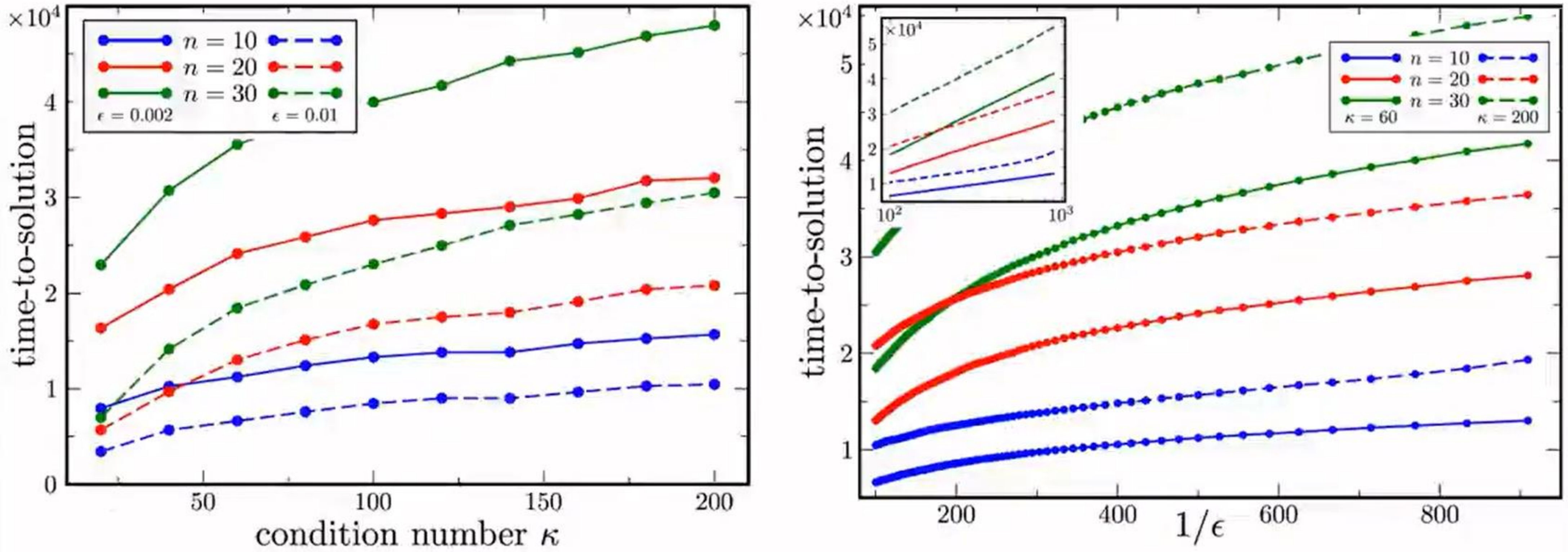
- Ising-type

$$A = \frac{1}{\zeta}\left(\sum_{j=1}^{n} \sigma_j^X + J \sum_{j=1}^{n-1} \sigma_j^Z \sigma_{j+1}^Z + \eta \mathbf{1}\right)$$

$$|b\rangle = H^{\otimes n}|0\rangle$$

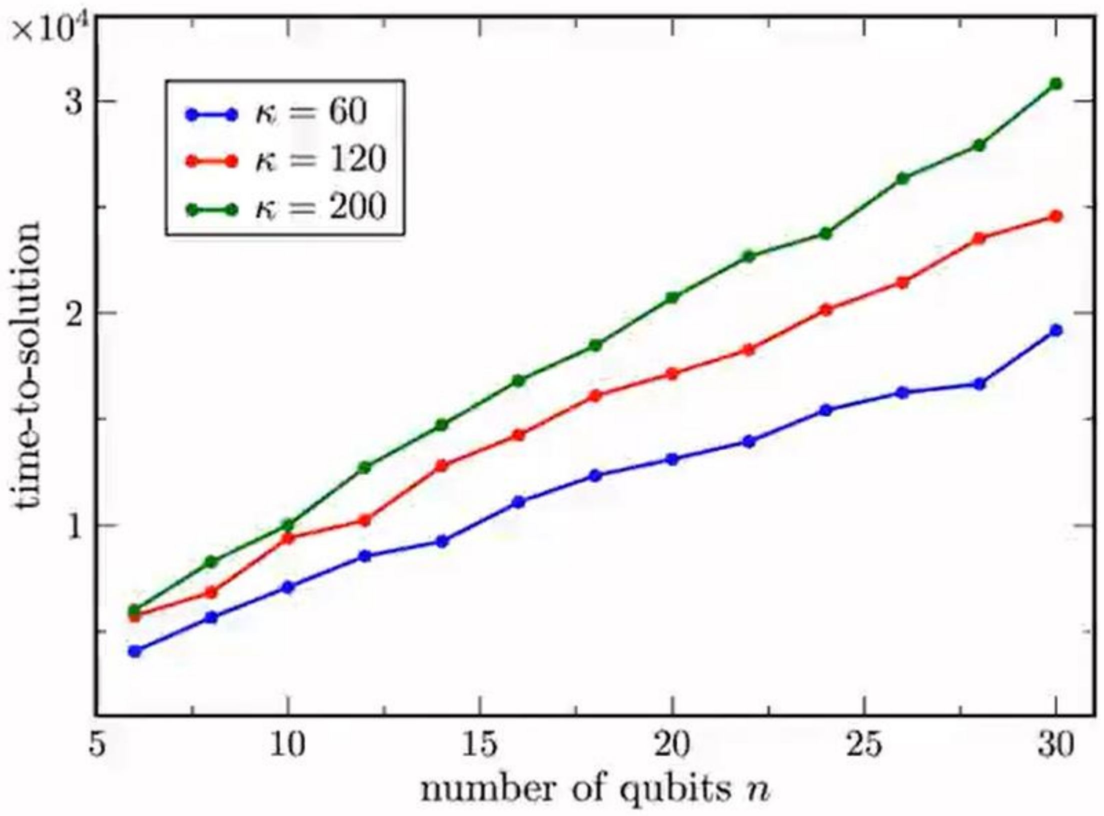
- ζ, η such that A has condition number κ

Example: scaling



- time-to-solution: number of iterations needed to **guarantee** precision ϵ
- sub-linear in κ
- logarithmic in $1/\epsilon$

Example: scaling



- time-to-solution: number of iterations needed to **guarantee** precision ϵ
- linear in n (logarithmic in N)

Example: simulations

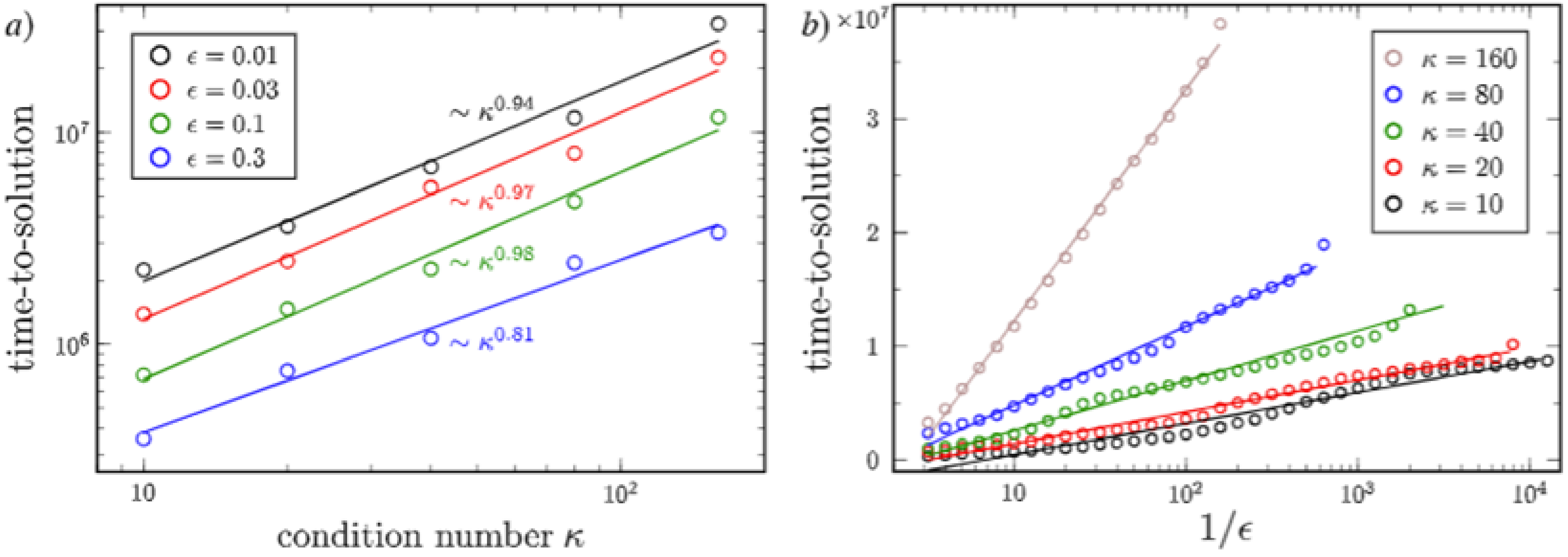
- random matrix

$$A = \frac{1}{\zeta}\left(\sum_j \sum_{k\neq j} p a_{j,k}\sigma_j^\alpha \sigma_k^\beta \ + \ \eta \mathbf{1}\right)$$

$$|b\rangle = H^{\otimes n}|0\rangle$$

- ζ, η such that A has condition number κ
- random:
  - $p \in \{0,1\}$
  - $a_{j,k} \in (-1,1)$
  - $\alpha, \beta \in \{X, Y, Z\}$

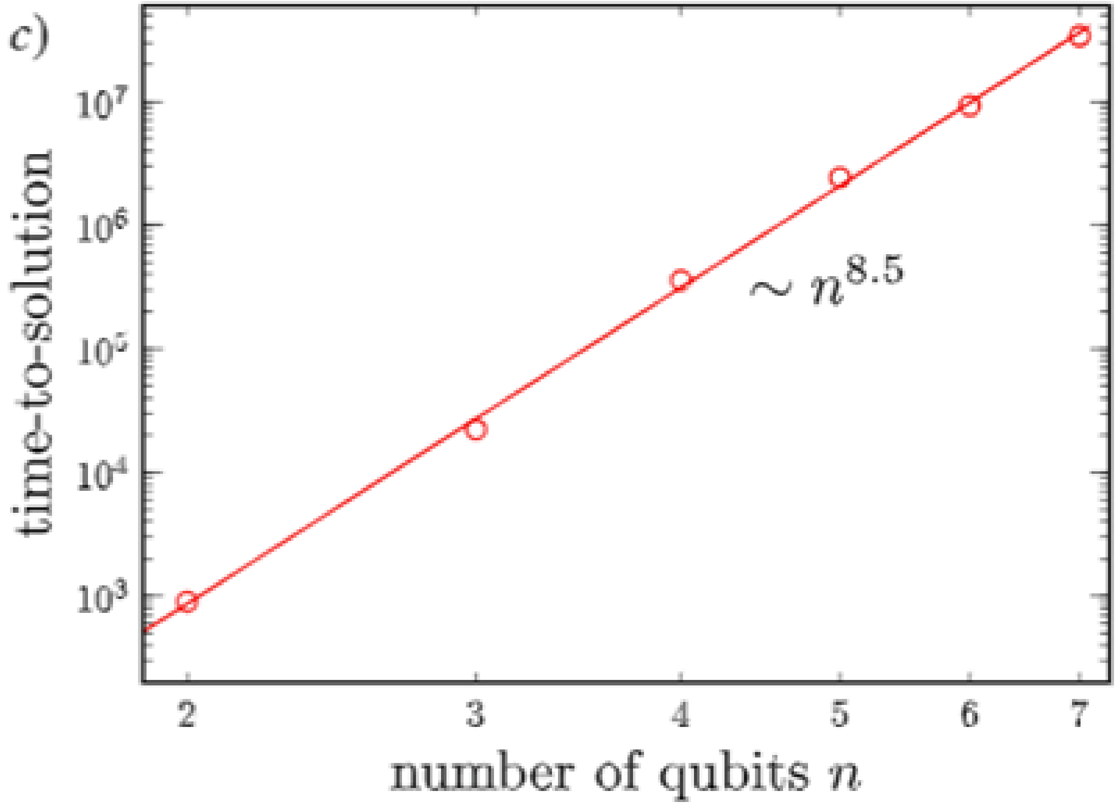
Example: scaling



- time-to-solution: number of iterations needed to **guarantee** precision ϵ
- slightly sub-linear in κ
- logarithmic in $1/\epsilon$

C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, arXiv:1909.05820

Example: scaling



- time-to-solution: number of iterations needed to **guarantee** precision ϵ
- polylogarithmic in N

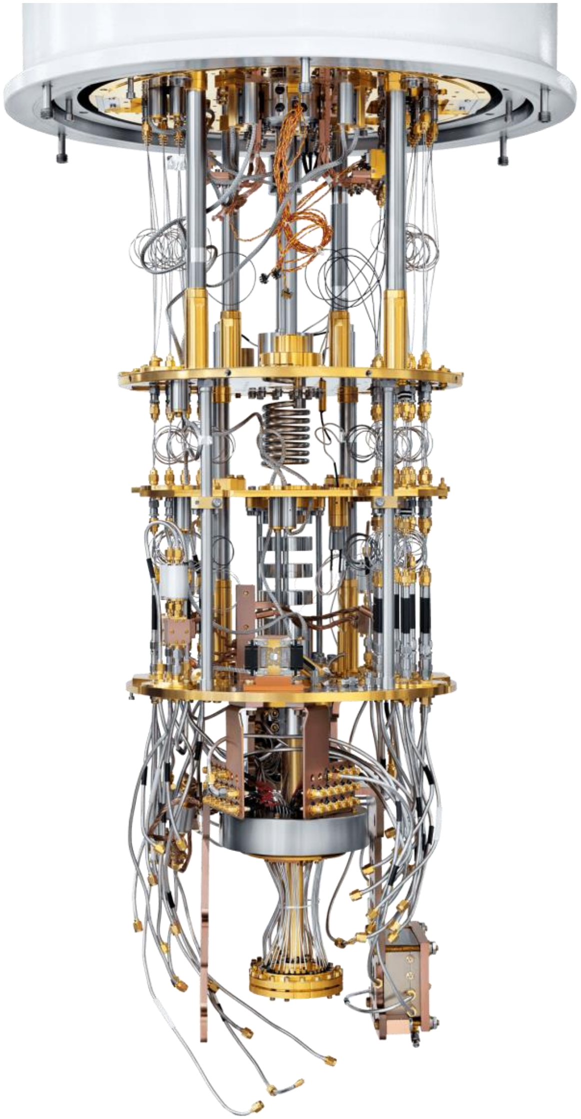
Example: Rigetti's quantum computer

- Ising-type

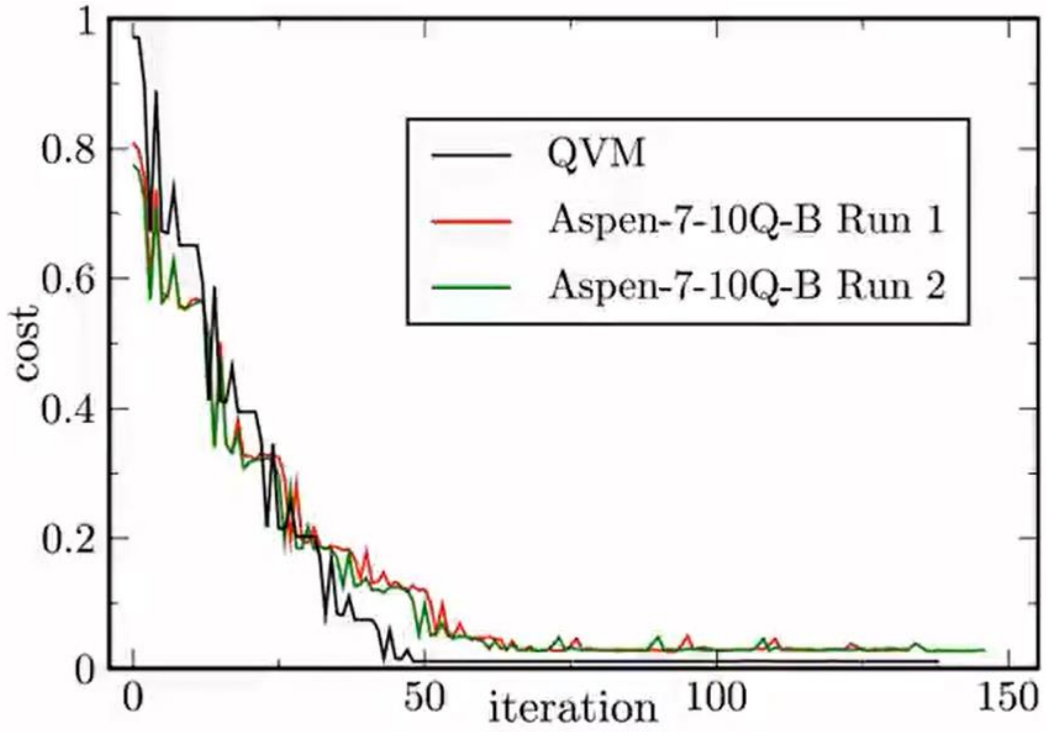
$$A = \frac{1}{\zeta}\left(\sum_{j=1}^{n} \sigma_j^X + J\sum_{j=1}^{n-1} \sigma_j^Z \sigma_{j+1}^Z + \eta \mathbf{1}\right)$$

$$|b\rangle = H^{\otimes n}|0\rangle$$

- ζ, η such that A has condition number κ



Example: Rigetti’s quantum computer



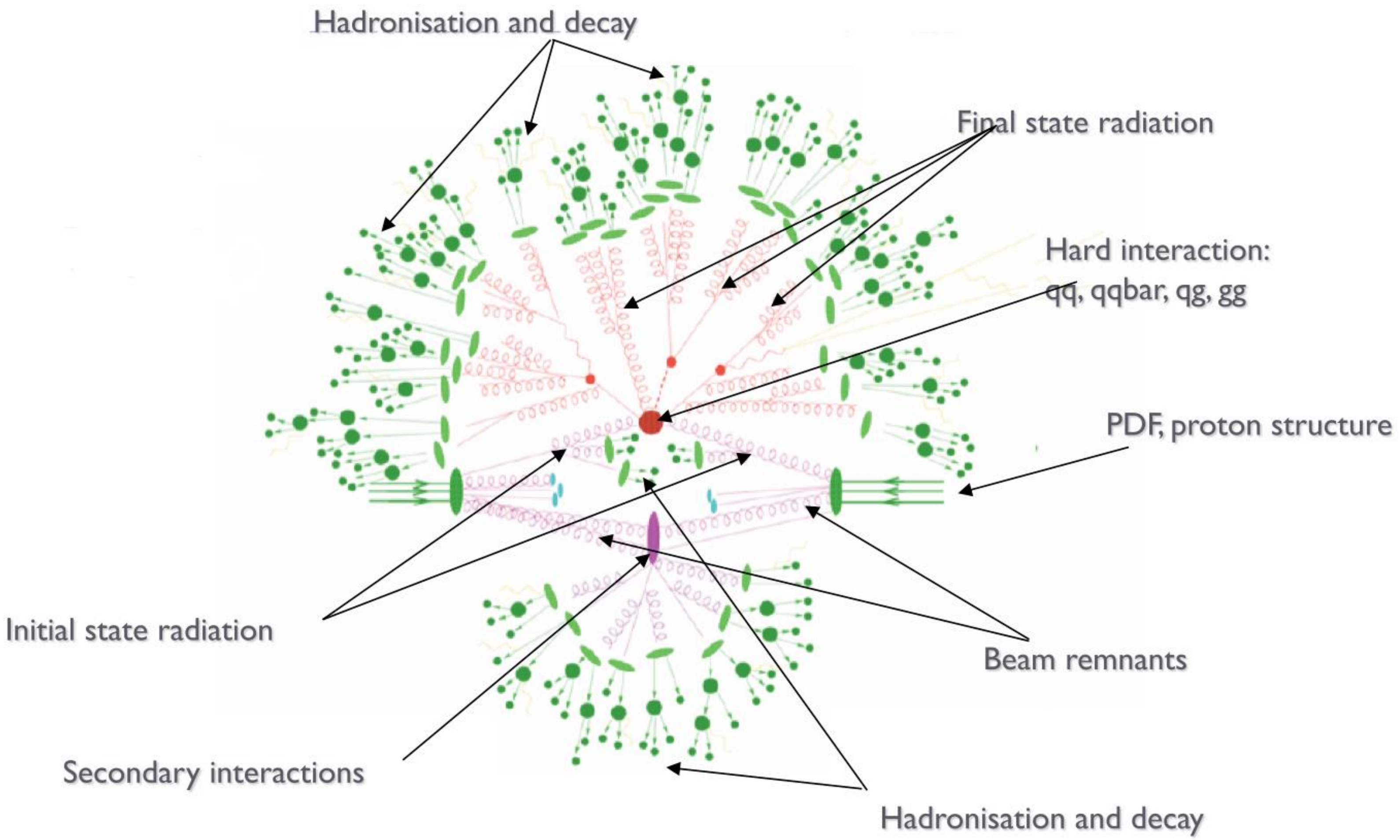
- largest implementation on real hardware: $n = 10$ qubits, 1024×1024
- noise resilience: correct parameters $\boldsymbol{\alpha}_{\text{opt}}$ despite cost $C > 0$

Style-based quantum generative adversarial networks for Monte Carlo events

with J. Baglio, M. Cè, A. Francis, D. M. Grabowska and S. Carrazza, arXiv:1909.05820

Context: Hadronic collisions at the LHC

LHC produces $O(10^9)$ proton collisions per second: huge complex environment



Simulation of the events are very intensive and requires lots of computing power

Machine learning approach to event generation

Since 2018, many papers have approached event generation with machine learning

THE EUROPEAN PHYSICAL JOURNAL C

Eur. Phys. J. C (2019) 79:4
https://doi.org/10.1140/epjc/s10052-018-6511-8

Regular Article - Theoretical Physics

Machine learning uncertainties with adversarial neural networks

Christoph Englert[1,a], Peter Galler[1,b], Philip Harris[2,c], Michael Spannowsky[3,d]

SciPost

SciPost Phys. 7, 075 (2019)

How to GAN LHC events

Anja Butter, Tilman Plehn and Ramon Winterhalder*

SciPost Physics **Submission**

MCNET-21-13

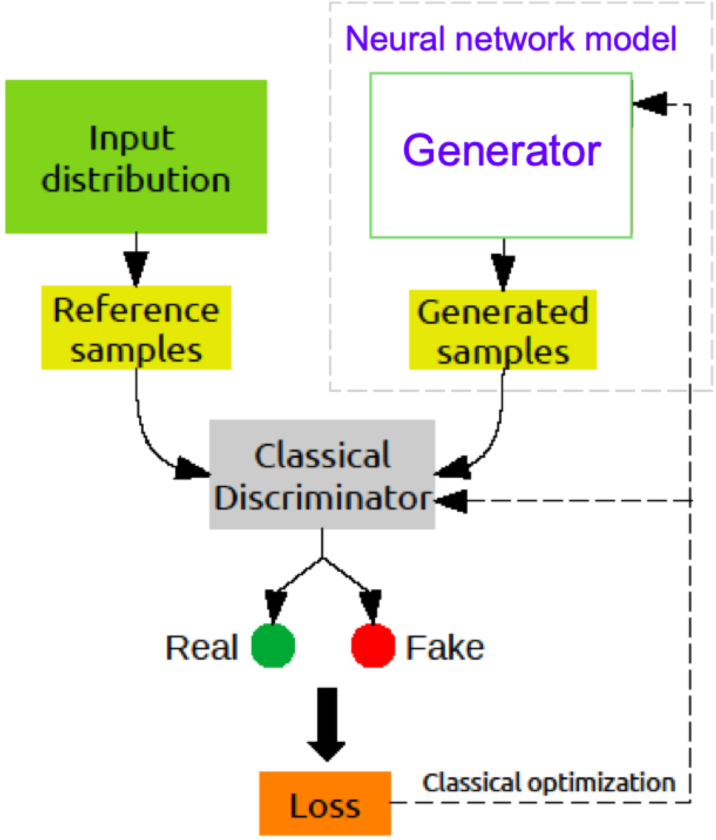
Accelerating Monte Carlo event generation – rejection sampling using neural network event-weight estimates

K. Danziger[1], T. Janßen[2], S. Schumann[2], F. Siegert[1]

Main idea: train with a small dataset, use machine learning networks to learn the underlying distribution and generate for free a much larger dataset

What is a generative adversarial network (GAN)?

Two networks competing: generator produces fake data, discriminator distinguishes between real (training) input data and fake (produced by the generator) data.
Adversarial game where the generator learns to map some input noise to the underlying (reference) distribution



Art forger analogy

Generator (art forger): Try creating fake paintings that look authentic.

Discriminator (art historian): Check paintings and try to catch the forgery.

Training: "Catch me if you can" game between the art forger and the art historian.

Success: Painted forgeries are so good that the art historian has at most a 50% guess ratio. The forger creates new work.

Training procedure

Training: Adapt alternatively the generator $G(\phi_g, z)$ and the discriminator $D(\phi_d, x)$

Mathematical tool: binary cross-entropy for the loss functions

- Generator loss function:

$$\mathcal{L}_G(\phi_g, \phi_d) = -\mathbb{E}_{z \sim p_{prior}(z)}[\log D(\phi_d, G(\phi_g, z))]$$

- Discriminator loss function:

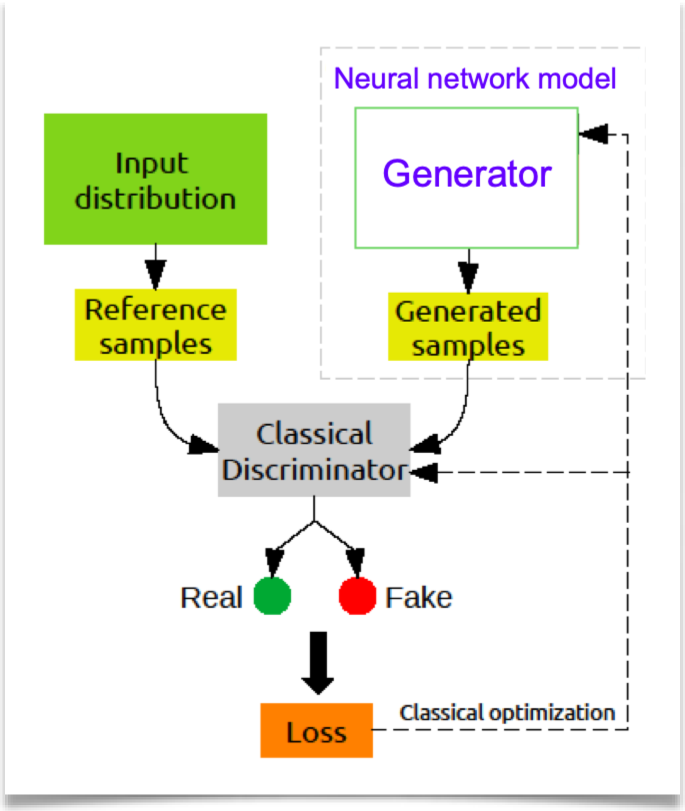
$$\mathcal{L}_D(\phi_g, \phi_d) = \mathbb{E}_{x \sim p_{real}(x)}[\log D(\phi_d, x)] + \mathbb{E}_{z \sim p_{prior}(z)}[\log(1 - D(\phi_d, G(\phi_g, z)))]$$

Game theory: min-max two-player game to reach Nash equilibrium

$$\min_{\phi_g} \mathcal{L}_G(\phi_g, \phi_d) \qquad \max_{\phi_d} \mathcal{L}_D(\phi_g, \phi_d)$$

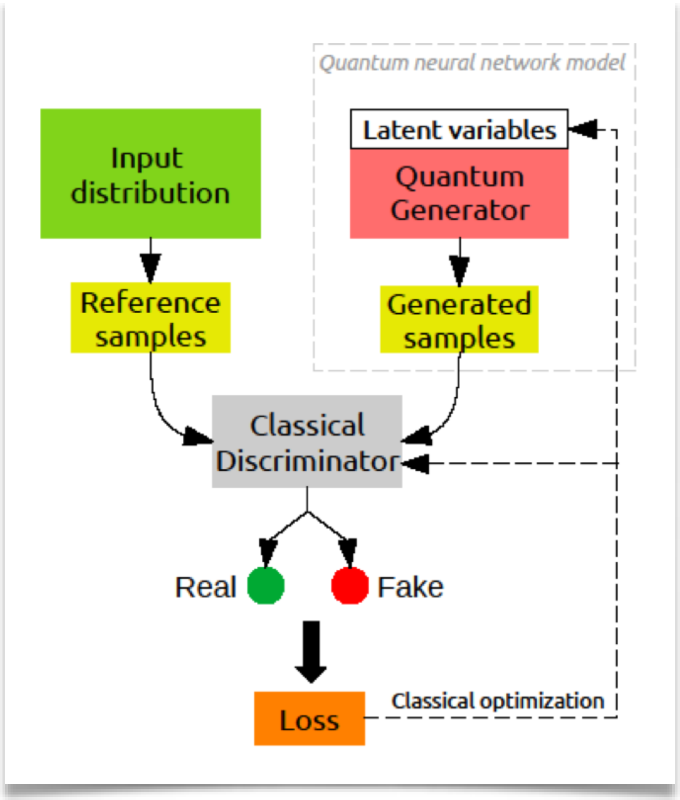
Hybrid approach for a qGAN

Classical setup:



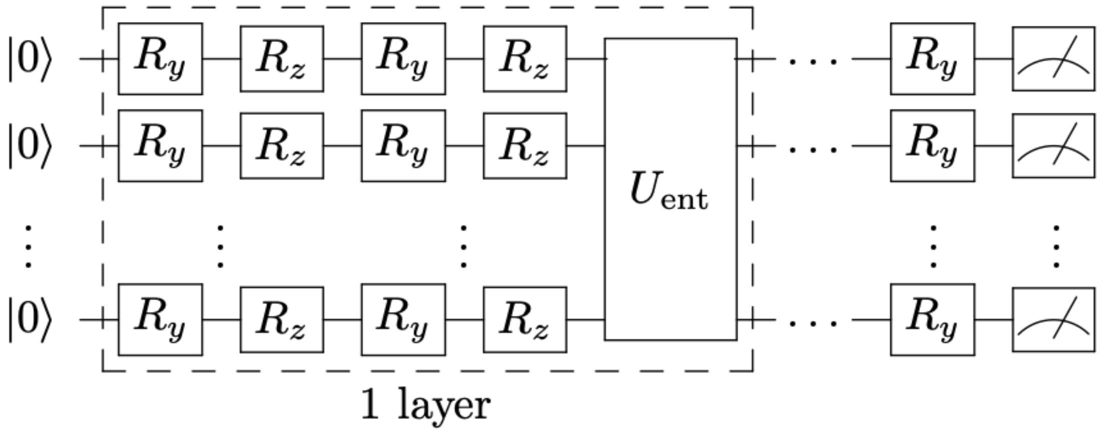
Only the generator becomes quantum

Hybrid quantum-classical setup:



Style-based quantum generator

Quantum generator: a series of quantum layers with rotation gates and entanglement operators



$$R_y = \exp\left(-i\frac{\theta}{2}\sigma_y\right),\ R_z = \exp\left(-i\frac{\theta}{2}\sigma_z\right)$$

U_{ent} set of controlled rotations for entanglement

1 component = 1 qubit

$$\vec{x}_{\text{fake}} = -\left[\langle\sigma_z^1\rangle, \langle\sigma_z^2\rangle, \ldots, \langle\sigma_n^1\rangle\right]$$

Style-based approach

Novelty of our network:
the noise is inserted in **every gate and not only in the initial quantum state**

$$R^i_{y,z}(\phi_g^{(i)}, \xi^{(j)}) = R_{y,z}(\phi_g^i \xi^j + \phi_g^{i+1})$$

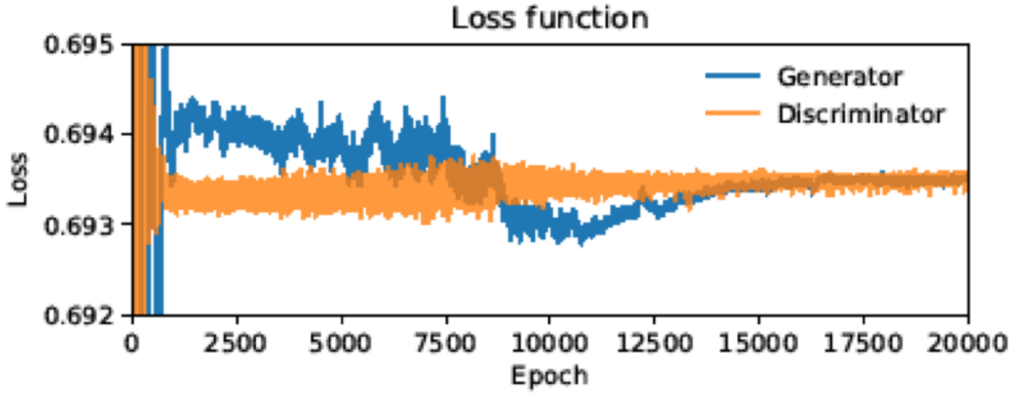
Circuit implemented in Python with Qibo [S. Efthymiou et al., arXiv:2009.01845] for quantum simulation

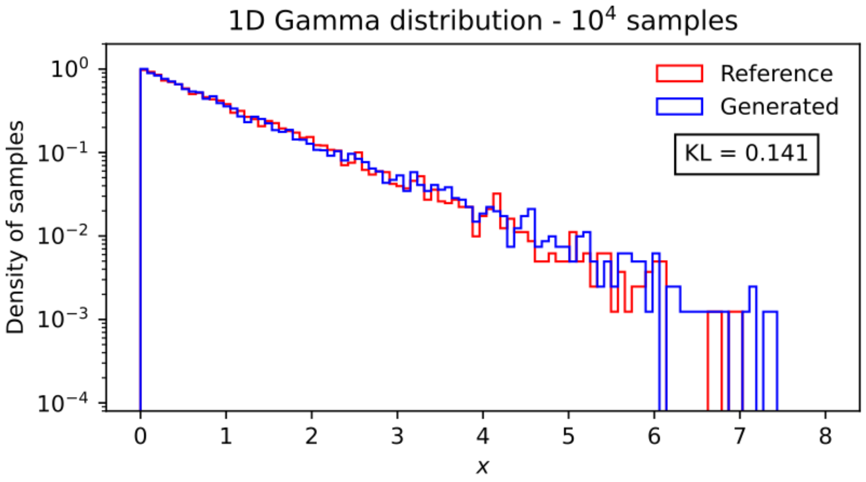
Validation: 1D Gamma distribution

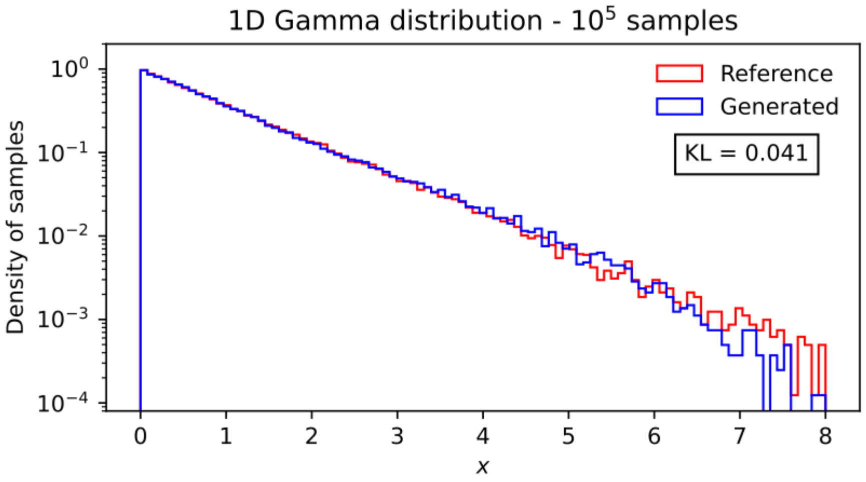
Assessing the validity of the approach: train and test on known distribution

With one qubit, **one layer**, using 100 bins: 1D Gamma function $p_\gamma(x,\alpha,\beta) = x^{\alpha-1}\frac{e^{-x/\beta}}{\beta^\alpha\Gamma(\alpha)}, \alpha = \beta = 1$

- Pre-processing of the data to fit samples in [-1,1]
- Train on 10^4 samples until convergence is reached, perform hyperparameter optimization
- Use generator to generate 10^4 and 10^5 samples to demonstrate reproducibility and data augmentation

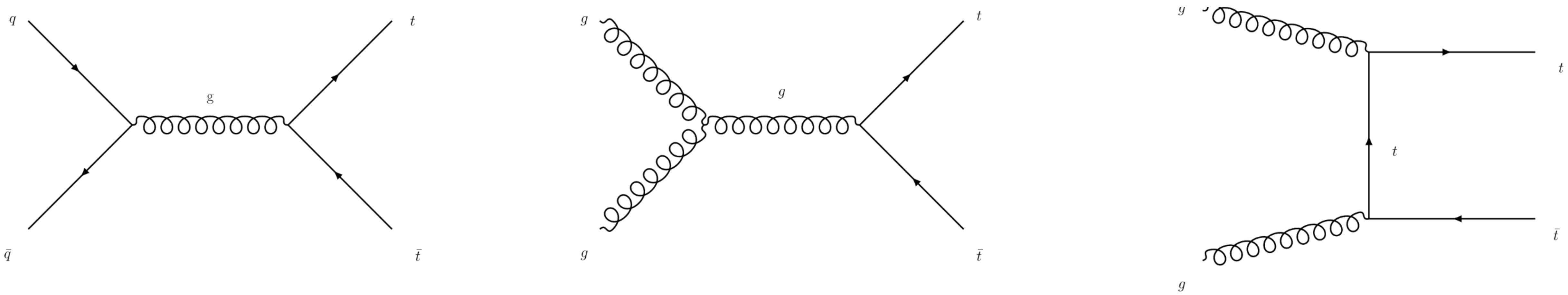






Simulation with actual LHC data

Testing the styled qGAN with real data: test-case with leading-order production $pp \to t\bar{t}$

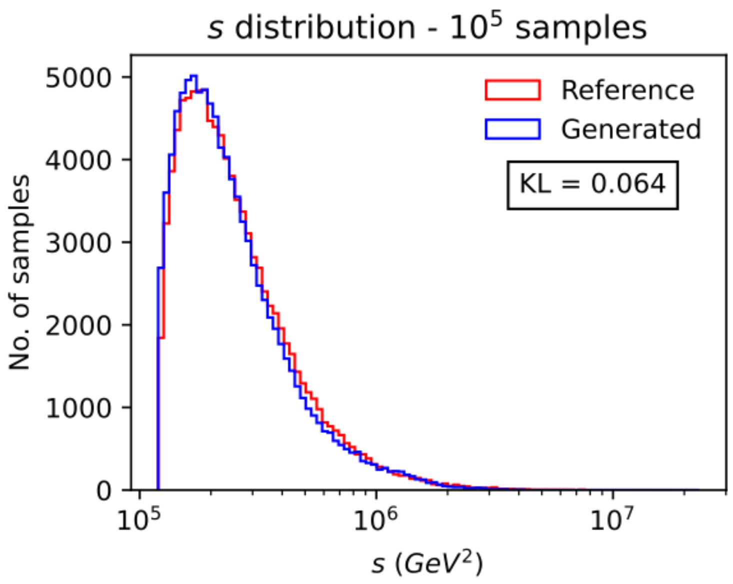


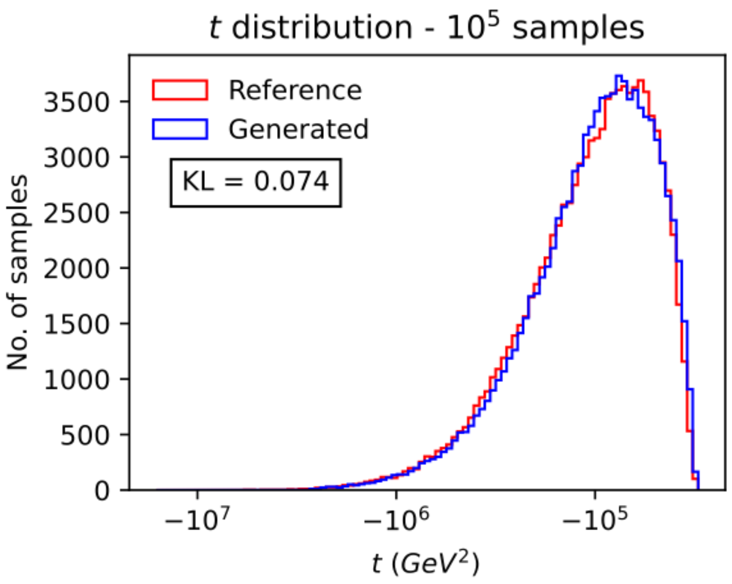
Training and reference samples generated with MadGraph5_aMC@NLO [Alwall et al., JHEP 07 (2014) 079]

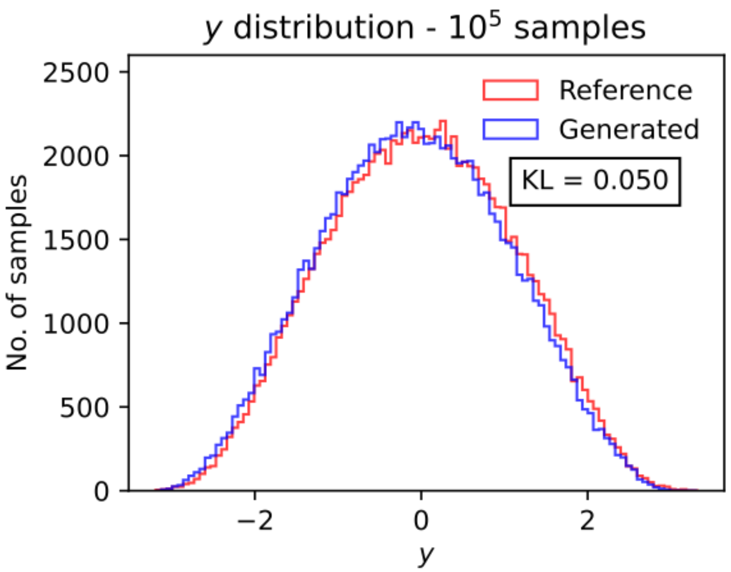
LHC at 13 TeV set-up, training set of 10^4 samples, Mandelstam variables (s, t) and rapidity y

Simulation with actual LHC data

After training, we assess the performance with simulations: 3 qubits, 2 layers, 100 bins

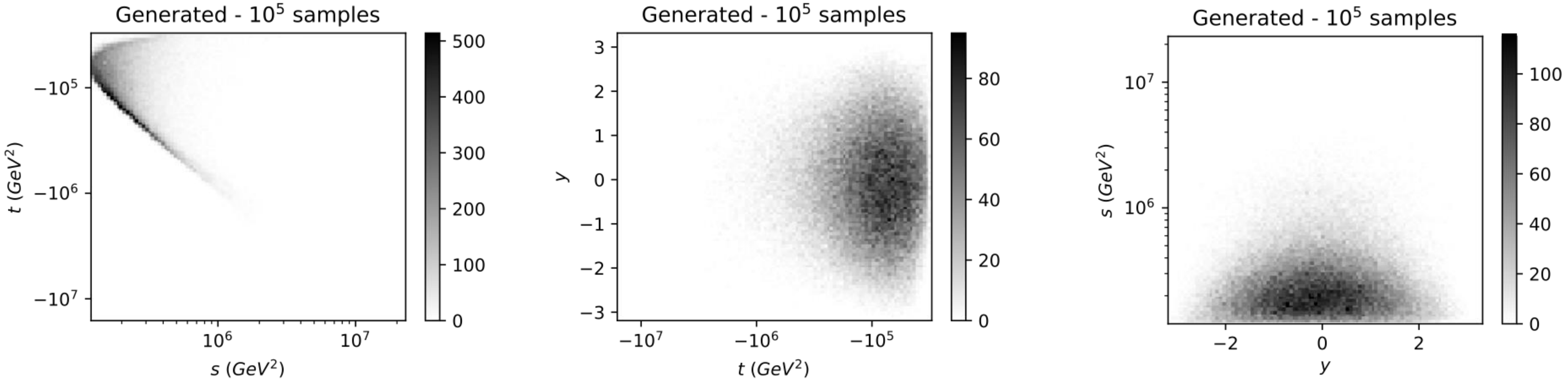






Simulation with actual LHC data

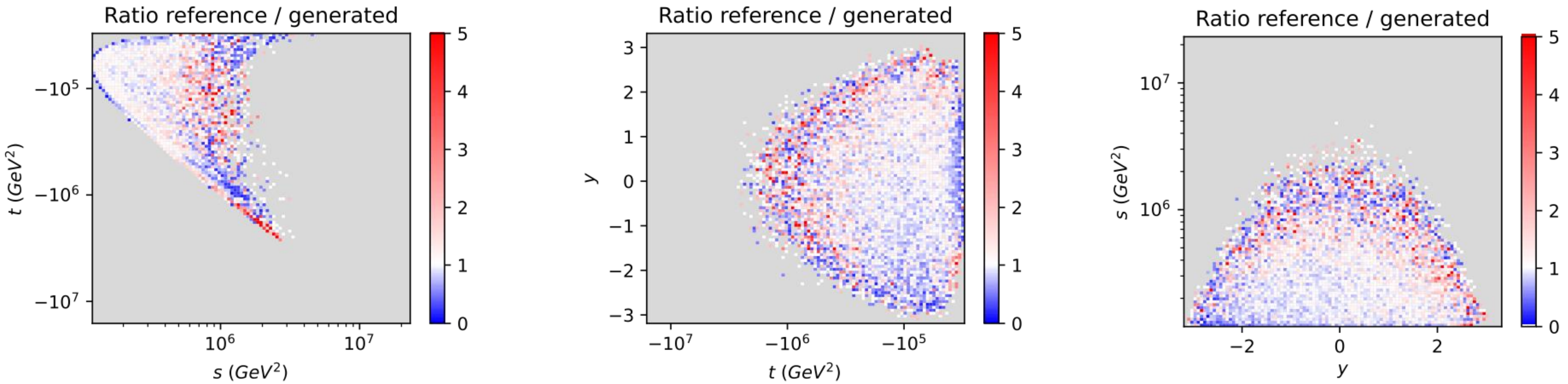
After training, we assess the performance with simulations: 3 qubits, 2 layers, 100 bins



Correlations are well captured!

Simulation with actual LHC data

After training, we assess the performance with simulations: 3 qubits, 2 layers, 100 bins



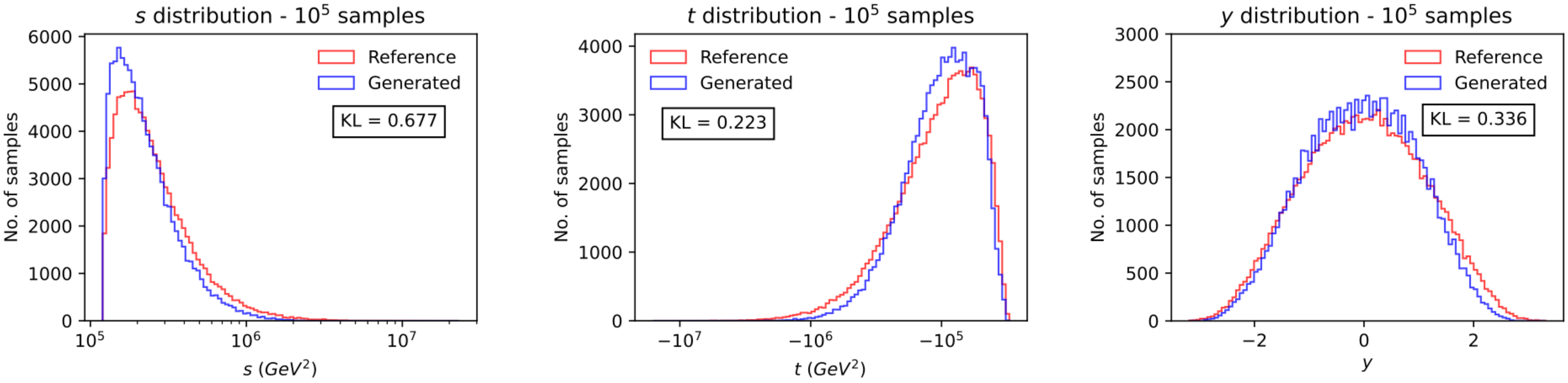
Remarkable low KL divergences with data augmentation!
Are these results maintained on real hardware ?

Results on IBM Q Hardware

Access to IBM quantum hardware via IBM Q cloud service
Technology of superconducting qubit

- Run on **ibmq_santiago 5-qubit machine**

Still good results with relatively low KL divergence!



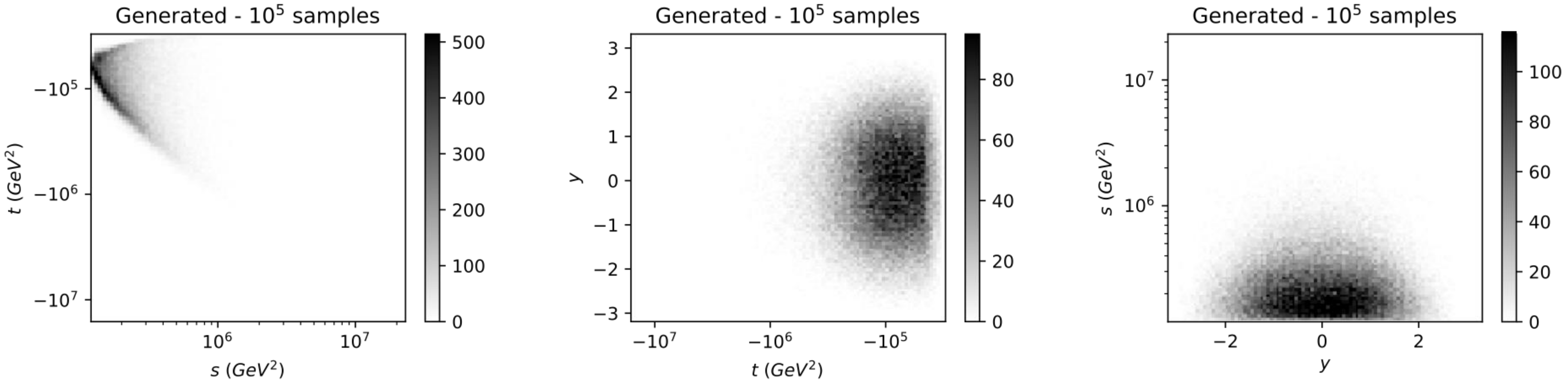
Results on IBM Q Hardware

Access to IBM quantum hardware via IBM Q cloud service

Technology of superconducting qubit

- Run on **ibmq_santiago 5-qubit machine**

Still good results with relatively low KL divergence!



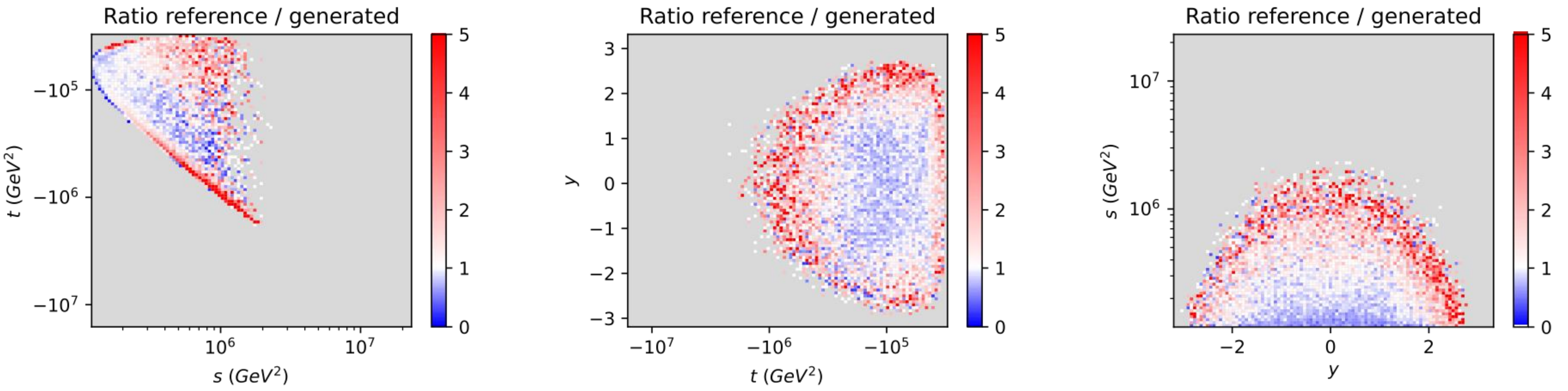
Results on IBM Q Hardware

Access to IBM quantum hardware via IBM Q cloud service

Technology of superconducting qubit

- Run on **ibmq_santiago 5-qubit machine**

Still good results with relatively low KL divergence!

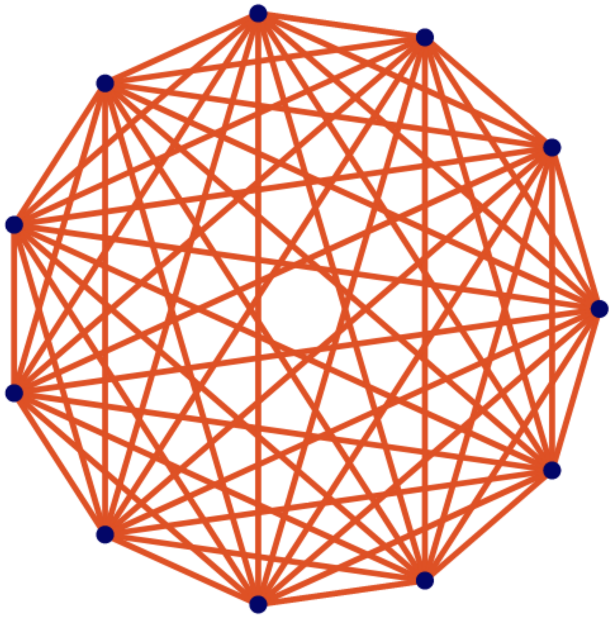


Testing different architectures

Superconducting transmon qubits: *ibmq_santiago* with 2-neighbouring site connectivity

Access via IBM Q cloud service

Trapped ion technology: ionQ with all-to-all connectivity



Access via Amazon Web Services

Testing different architectures: results

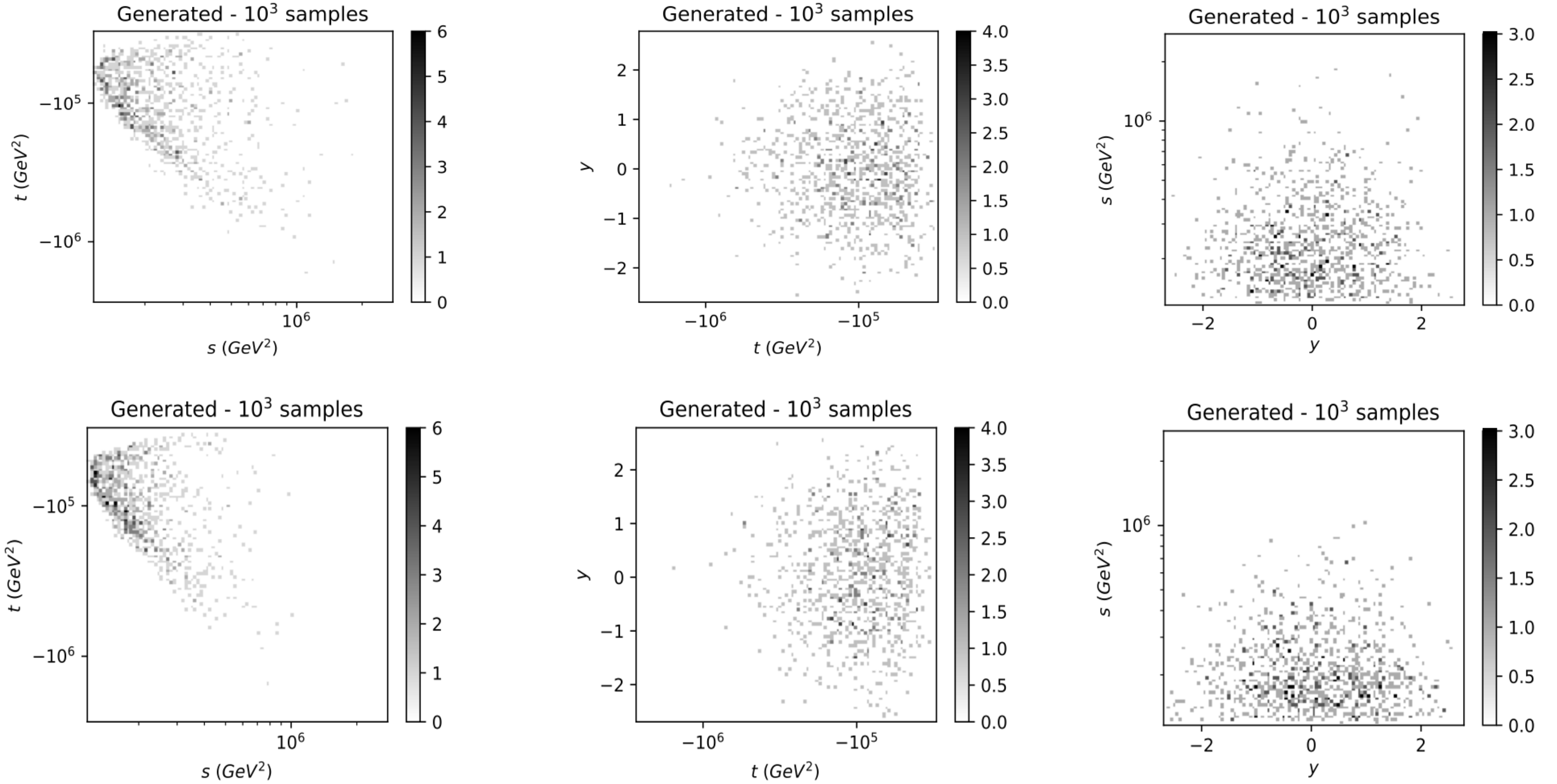
- Access constraints to ionQ: test limited to 1k samples only

Very similar results:

implementation largely hardware-independent

ionQ samples:

IBM Q samples:



Thanks for your attention

Any questions?