# On Some of UNIBO's Research Contributions to Spoke 10

**Ugo Dal Lago** **Elisa Ercolessi**

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

*QCSW, October 12th 2023*

# Circuit Width Estimation via Index Refinement Types

Ugo Dal Lago – ugo.dallago@unibo.it

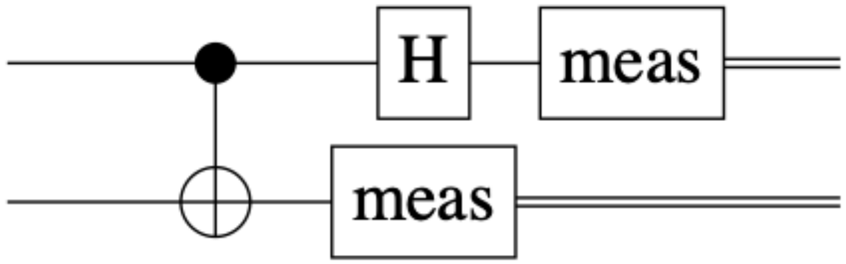
Joint work with Andrea Colledan – andrea.colledan@unibo.it

October 12, 2023

# (Quantum) Circuit Description Languages

- *Classical* programs that *describe* quantum computations
- Qiskit, Cirq, **Quipper** [1], …

```
alice :: Qubit -> Qubit -> Circ (Bit,Bit)
    alice q a = do
    a <- qnot a ‘controlled‘ q
    q <- hadamard q
    (x,y) <- measure (q,a)
    return (x,y)
```



# Problems [2]

| Circuit description Languages | vs | Real-world quantum computers |
|---|---|---|
| • High level | | • Low level |
| • Millions of qubits | | • A couple hundred qubits |
| • Trillions of gates | | • Less than a thousand gates |

Idea: formal methods

# Refinement Types

- Refinement type = regular type + predicate [3,4]

```
{n : Int | n >= 1 && n <= 12}
```

- **Index refinement type** = regular type + *index* [5,6]

```
Int[1,12], List[5] Int
```

- Support for limited *dependency*

```
List[i] Int -> List[j] Int -> List[i+j] Int
```

# Proto-Quipper-R (PQR) Overview

- Extending the Proto-Quipper [7] calculus with refinements

- $\mathsf{Circ}^I\ (T,U)$ : Circuits of width *at most* $I$
- $A \xrightarrow{I} B$ : Functions that build a circuit of width *at most* $I$
- $\mathsf{List}^I\ A$ : Lists of length *exactly* $I$

- $I,J ::=$ ... | $n$ | $i$ | $I + J$ | $\max(I,J)$
- $V,W ::=$ ... | $\ell$ | $(\ell,C,\ell')$ | nil | cons $V\ W$
- $M,N ::=$ ... | $\mathsf{apply}(V,W)$ | $\mathsf{fold}_i\ V\ W$

# PQR Type System

$$\Theta;\Gamma;Q \vdash M : A \ ; I$$

"For all index variables in $\Theta$, under typing contexts $\Gamma$ and $Q$, $M$ is a program of type $A$ and builds a circuit of width at most $I$."

- Subtyping judgment: $\Theta \vdash_s A <: B$
- Semantic relationship between indices: $\Theta \vDash I \leq J$
- Ideally delegated to an **external SMT solver** [8]

# Quantum Fourier Transform in PQR

$$qft \triangleq \mathsf{fold}_i\ \ qftStep\ \mathsf{nil}$$

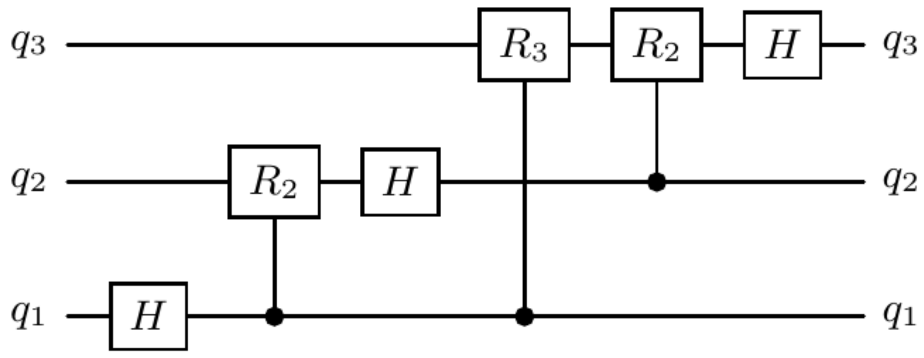
$$\begin{aligned}
qftStep \triangleq\ & \mathsf{lift}(\mathsf{return}\ \lambda\langle qs, q\rangle_{\mathsf{List}^i\ \mathsf{Qubit}\otimes\mathsf{Qubit}}.\\
& \quad \mathsf{let}\ \langle n, qs'\rangle = qlen\ qs\ \mathsf{in}\\
& \quad \mathsf{let}\ revQs' = rev\ qs'\ \mathsf{in}\\
& \quad \mathsf{let}\ \langle q', qs''\rangle = (\mathsf{fold}_j\ \ (\mathsf{lift}(rotate\, n))\ \langle q, \mathsf{nil}\rangle)\ revQs'\ \mathsf{in}\\
& \quad \mathsf{let}\ q'' = \mathsf{apply}(\mathsf{H}, q')\ \mathsf{in}\\
& \quad \mathsf{return}\ (\mathsf{cons}\ q''\ \ qs''))
\end{aligned}$$

$$\begin{aligned}
rotate \triangleq\ & \lambda n_{\mathsf{Nat}}.\mathsf{return}\ \lambda\langle\langle q, cs\rangle, c\rangle_{(\mathsf{Qubit}\otimes\mathsf{List}^i\ \mathsf{Qubit})\otimes\mathsf{Qubit}}.\\
& \quad \mathsf{let}\ \langle m, cs'\rangle = qlen\ cs\ \mathsf{in}\\
& \quad \mathsf{let}\ rgate = makeRGate\,(n+1-n)\ \mathsf{in}\\
& \quad \mathsf{let}\ \langle c', q'\rangle = \mathsf{apply}(rgate, \langle c, q\rangle)\ \mathsf{in}\\
& \quad \mathsf{return}\ \langle q', \mathsf{cons}\ c'\ \ cs'\rangle
\end{aligned}$$

$$qft\ ::\ \mathsf{List}^i\ \mathsf{Qubit} \xrightarrow{\ i\ } \mathsf{List}^i\ \mathsf{Qubit}$$

$$qft\ ::\ \mathsf{List}^3\ \mathsf{Qubit} \xrightarrow{\ 3\ } \mathsf{List}^3\ \mathsf{Qubit}$$

$$qft\,(\mathsf{cons}\ q_1\ (\mathsf{cons}\ q_2\ (\mathsf{cons}\ q_3\ \mathsf{nil})))\ ::\ \mathsf{List}^3\ \mathsf{Qubit}\ ;\ 3$$



# Conclusion & Future Work

- The language is defined and has been proven to be
  - Type-safe: evaluation preserves types
  - **Correct:** the derived upper bounds actually hold at runtime
- The language can describe and verify a realistic algorithm (i.e. QFT)

- Future work:
  - Generalizing to other kinds of resource consumption
  - Automatizing the type-checking process
  - Implementation and interaction with SMT solvers

# Bibliography


[1] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, “Quipper: a scalable quantum programming language,” in Proc. of PLDI, 2013.

[2] Z. Yang, M. Zolanvari, and R. Jain, “A Survey of Important Issues in Quantum Computing and Communications,” IEEE Commun. Surv. Tutorials, vol. 25, no. 2, 2023.

[3] P. M. Rondon, M. Kawaguci, and R. Jhala, “Liquid types,” in Proc. of PLDI, 2008.

[4] N. Vazou, E. L. Seidel, R. Jhala, D. Vytiniotis, and S. Peyton-Jones, “Refinement types for Haskell,” in Proc. of ICPF, 2014.

[5] U. D. Lago and M. Gaboardi, “Linear Dependent Types and Relative Completeness,” Logical Methods in Computer Science, vol. 8, Issue 4, 2012.

[6] E. Çiçek, D. Garg, and U. Acar, “Refinement Types for Incremental Computational Complexity,” in Programming Languages and Systems, vol. 9032, J. Vitek, Ed., in Lecture Notes in Computer Science, vol. 9032, 2015.

[7] F. Rios and P. Selinger, “A Categorical Model for a Quantum Circuit Description Language (Extended Abstract),” Electron. Proc. Theor. Comput. Sci., vol. 266, pp. 164–178, Feb. 2018.

[8] A. Biere, M. Heule, and H. van Maaren, Handbook of Satisfiability. Amsterdam, NY: IOS Press, 2009.

[9] A. Colledan, U. Dal Lago, "On Dynamic Lifting and Effect Typing in Circuit Description Languages," in TYPES 2022, 2023.

# Quantum algorithm on NISQ devices

- For solution of both classical and quantum applications
- State preparation of non-trivial problems

Elisa Ercolessi

# DYNAMICS OF LATTICE GAUGE THEORIES

**ENCODING of the STATES**
$|\psi\rangle$ as q-bits

⬇

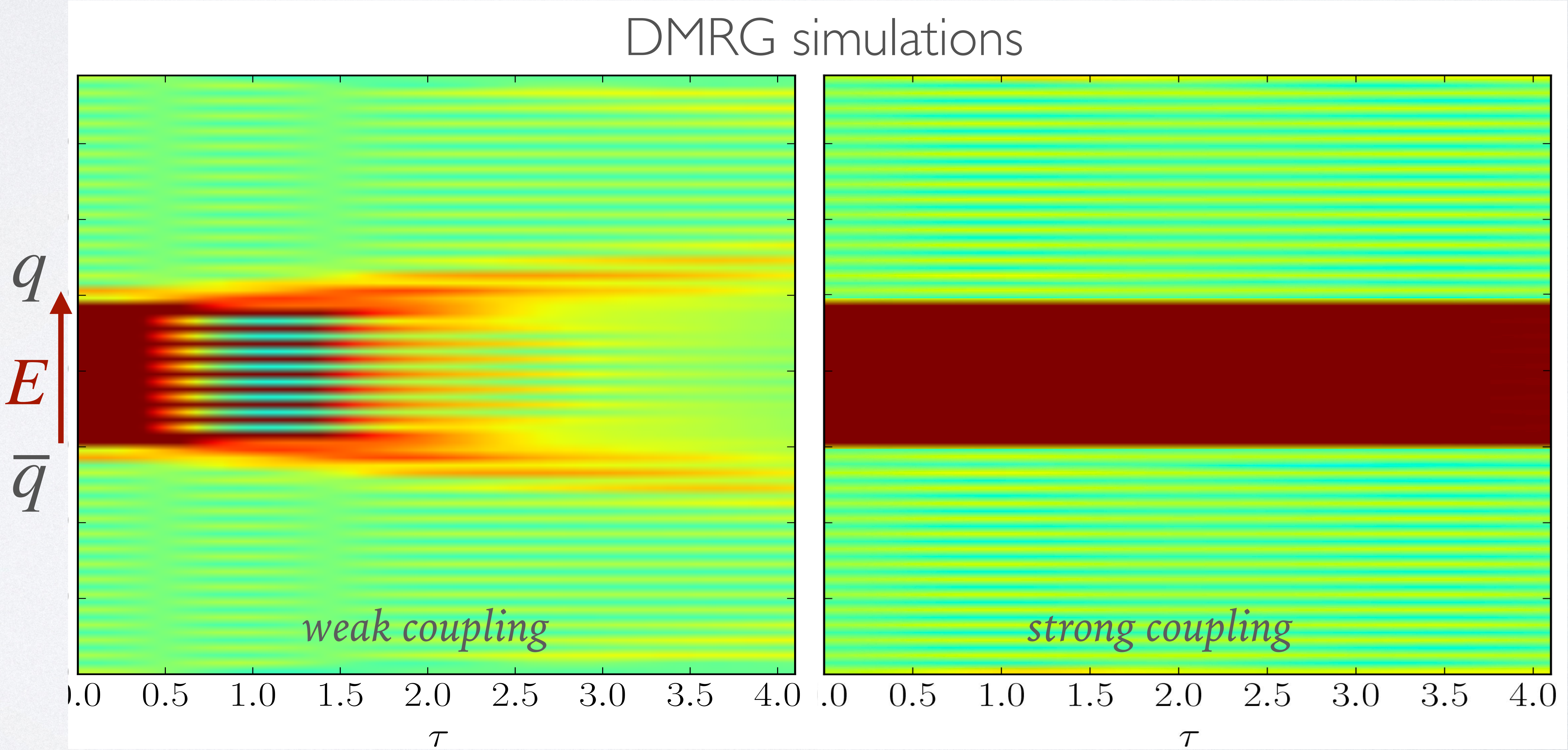
**QUANTUM EVOLUTION**
$e^{-itH}|\psi\rangle$ as q-gates

⬇

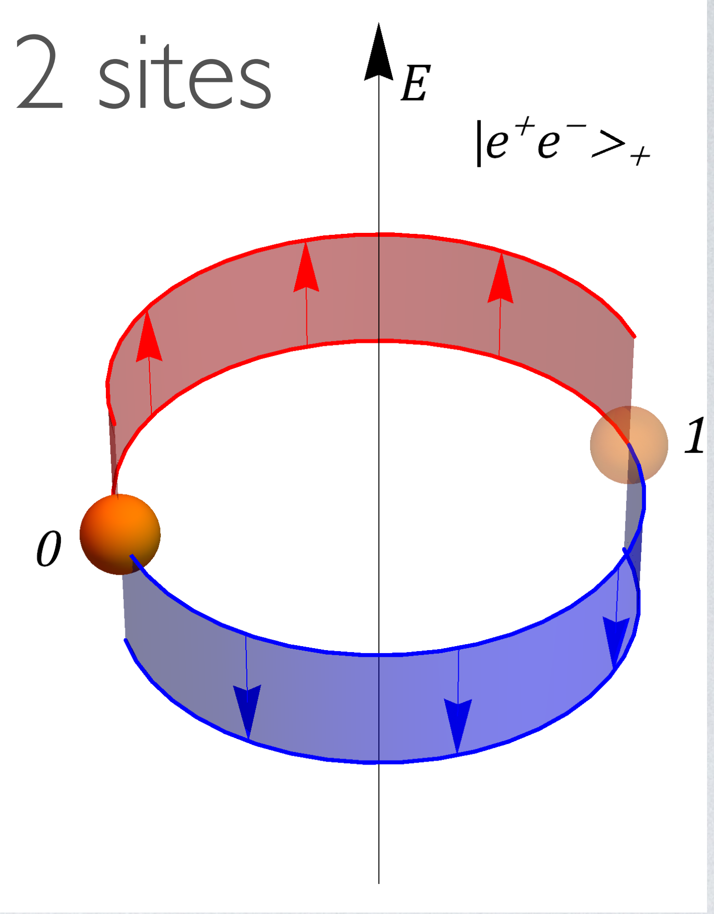
**MEAN VALUES & CORRELATORS**
from q-measurements

➤ STRING BREAKING

in the one-dimensional $\mathbb{Z}_n$ Schwinger model



# On **NISQ device:**




**Positive parity states can be represented with just 2 qubits**

$\#$ states 36 $\xrightarrow{\text{Gauss Law}}$ # physical states 6 $\xrightarrow{\text{Parity}}$ # $\mathcal{P}^+$ states 4

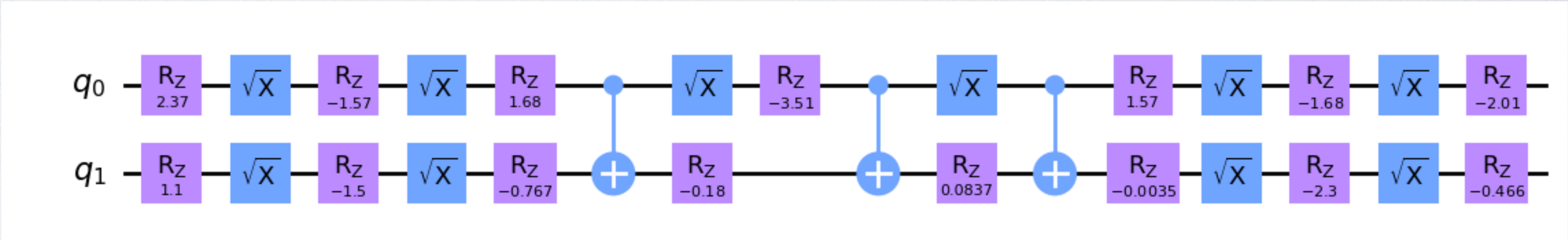
- embedding:

Hilbert space reduction
thanks to Gauge symmetries and Parity

- evolution:

via Trotter

- fixed depth for 2 sites
- minimisation of 3 qubit gates

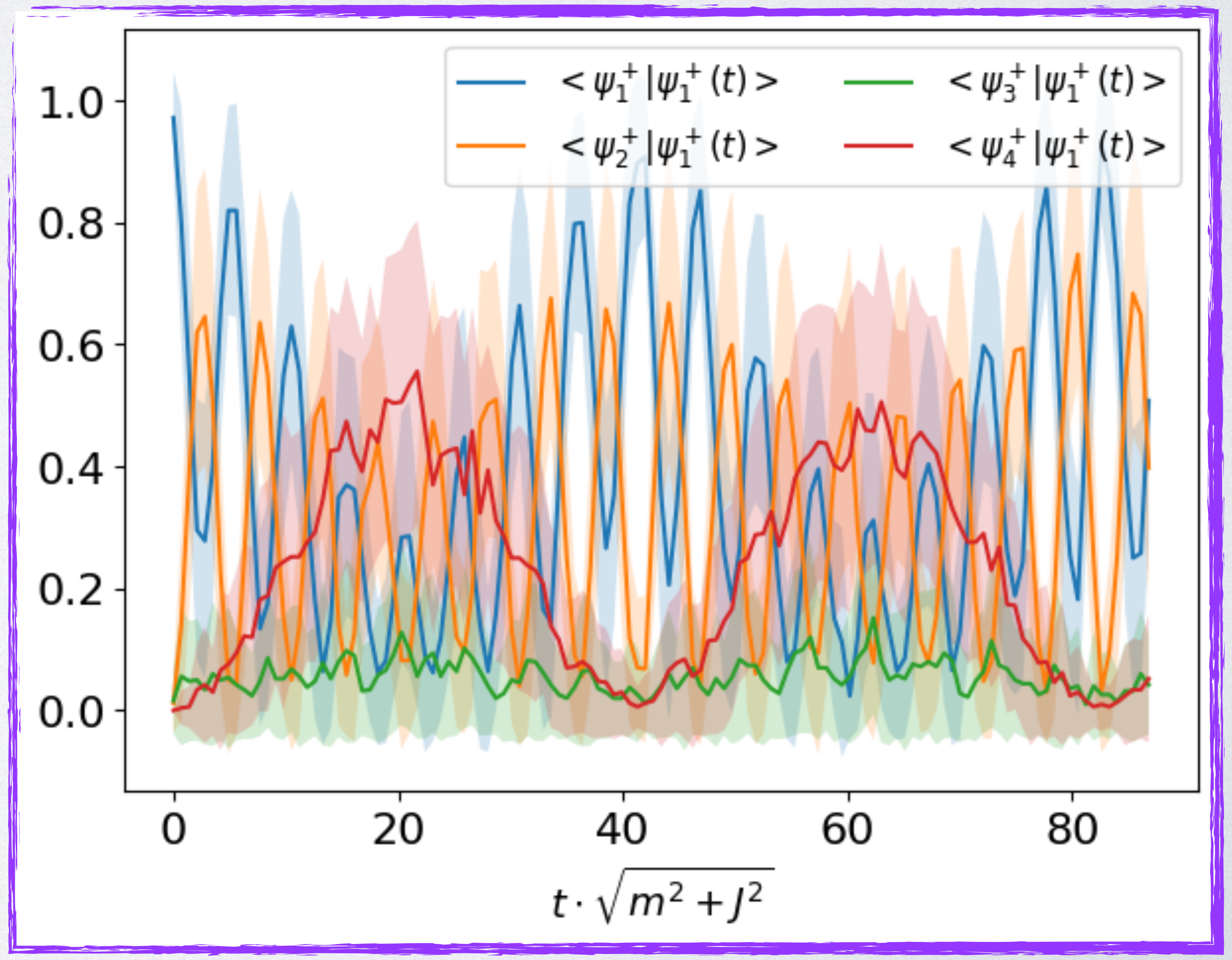



- measure:

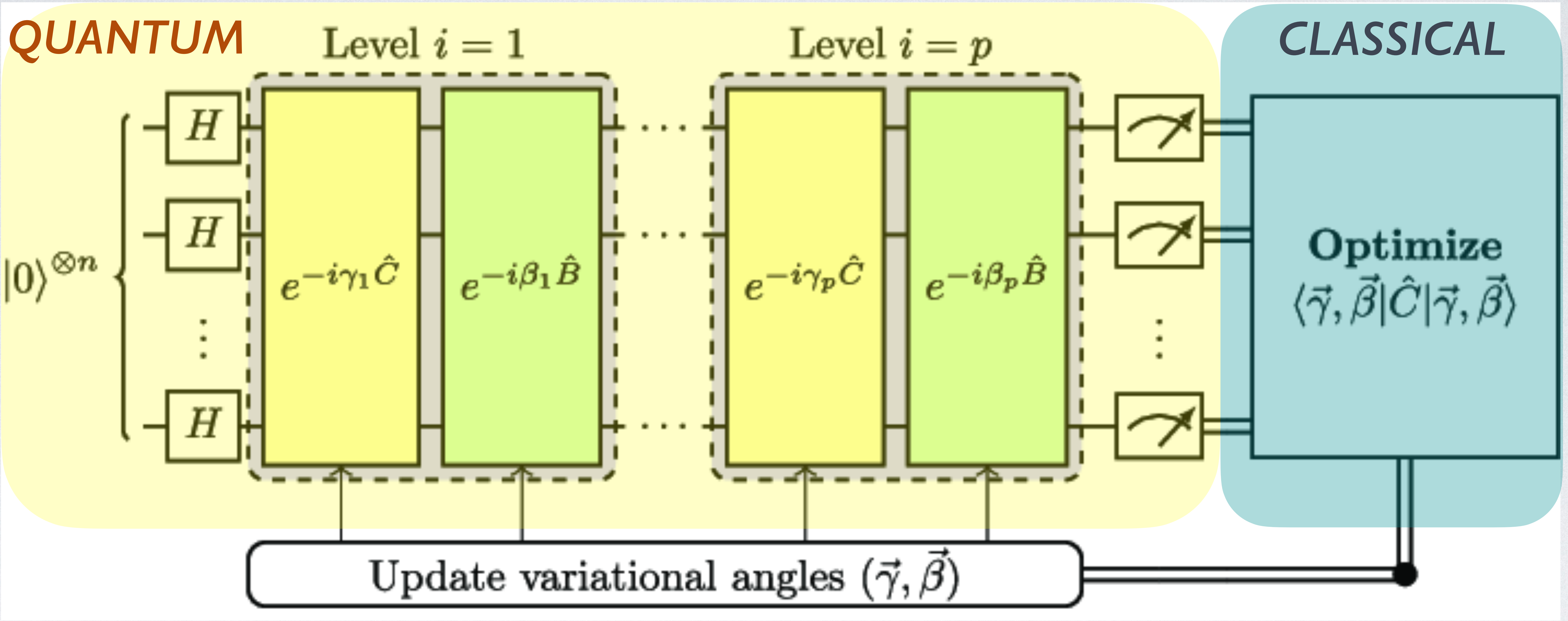
of Loschmid amplitude after a quench

Noisy Simulation
(including SPAM. Doppler damping, amplitude)
on PASQAL Pulser SDK

➤ **signals of Dynamical Phase Transitions**

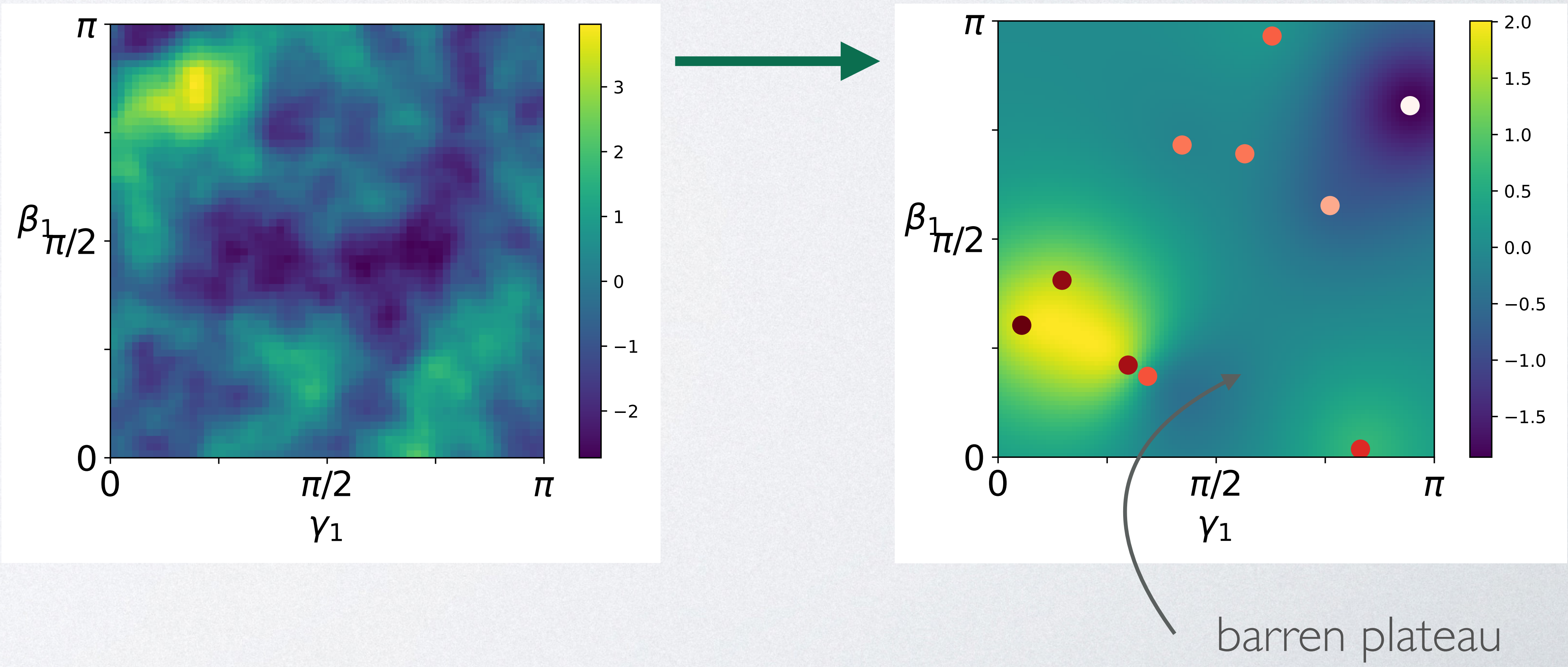


# BAYESIAN OPTIMIZATION FOR QAOA

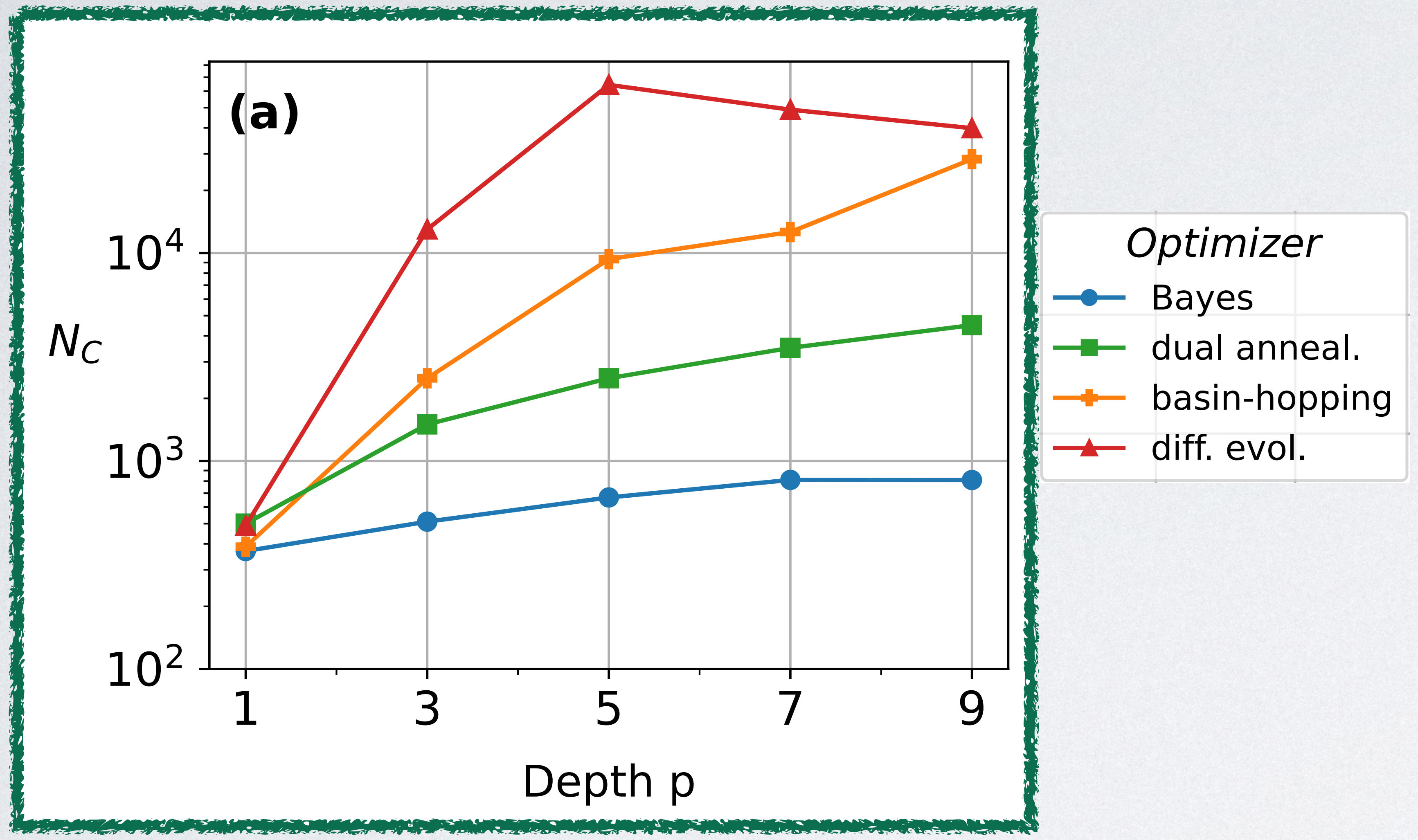


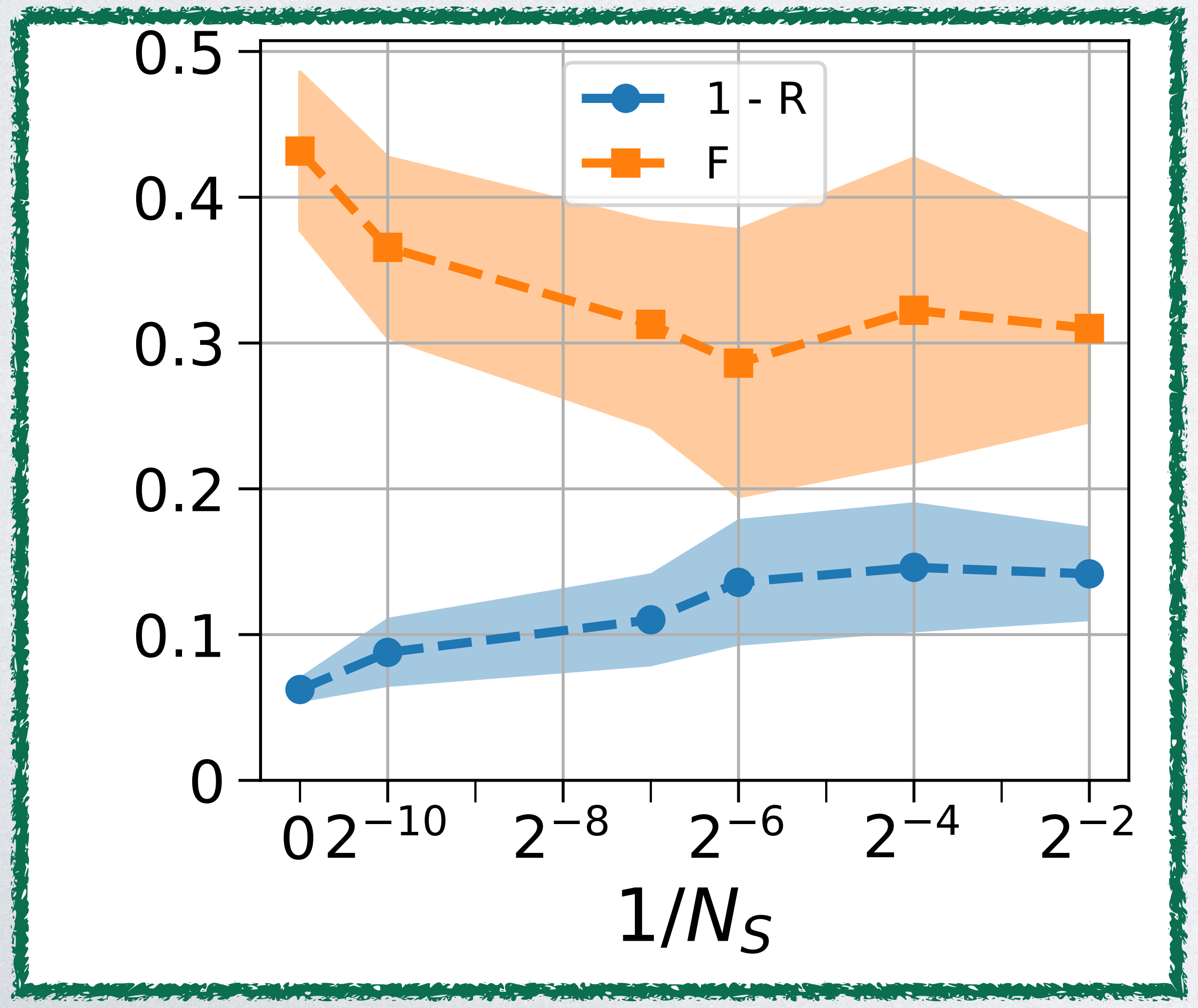

- We use a stochastic Gaussian Process (GP) to recreates the landscape of a function
- Bayesian Optimization exploits the prediction of the GP to propose a new set of optimal parameters at every step of QAOA

cost function reconstruction

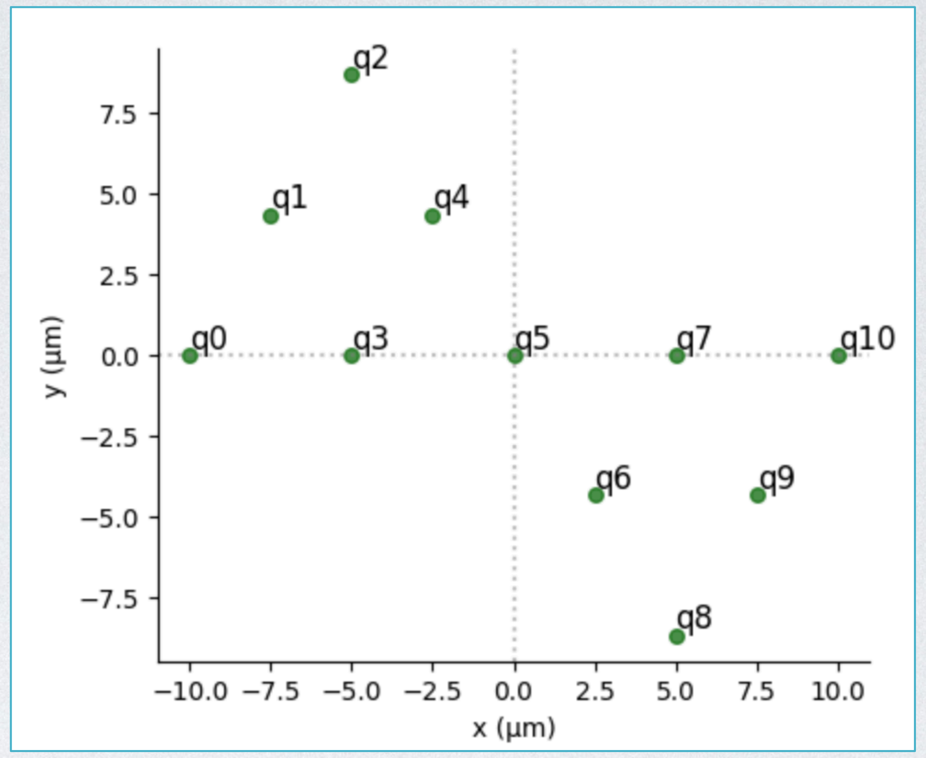


to solve a number of issues:
number of points in landscape
number of measurements
noisy circuits

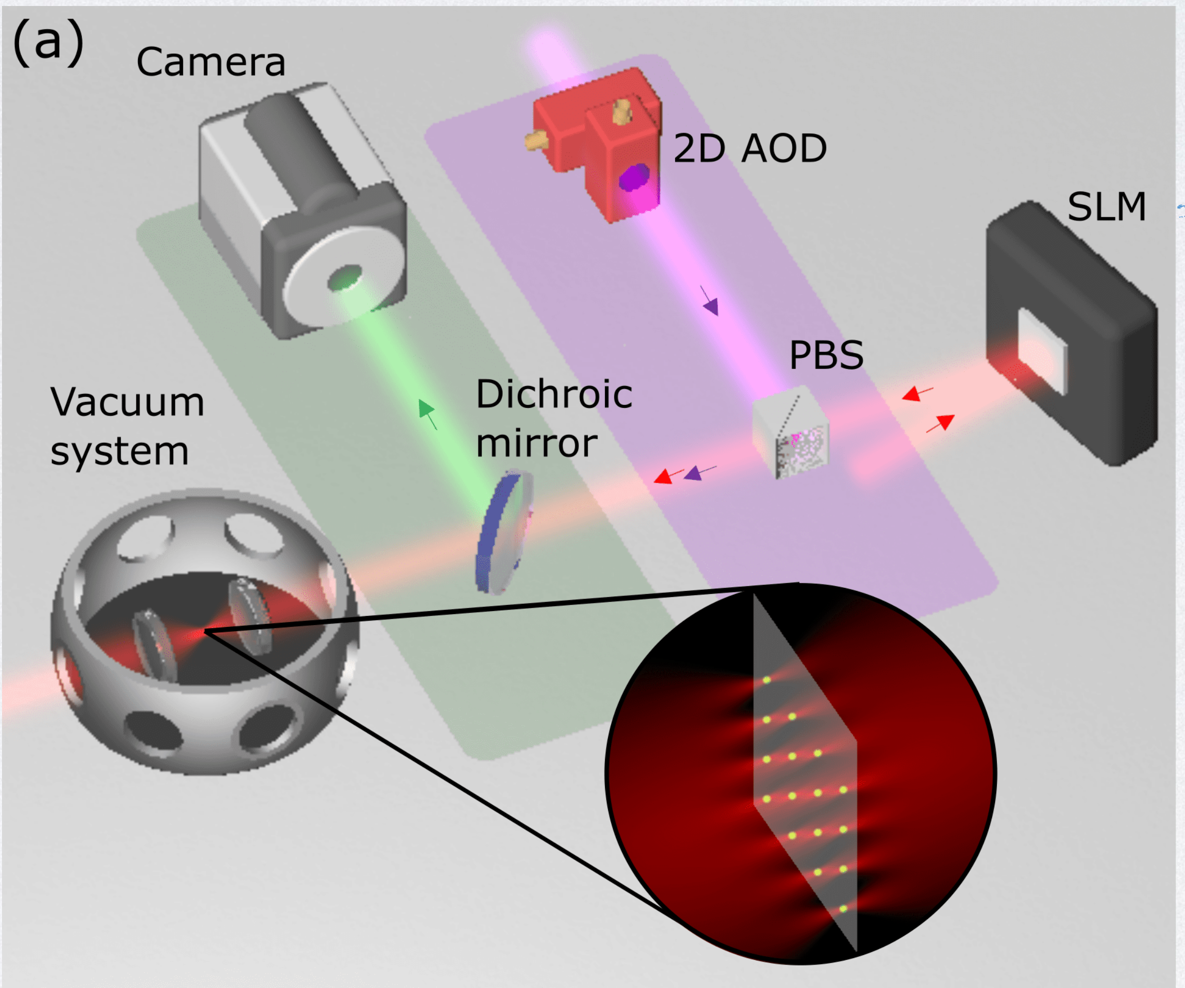


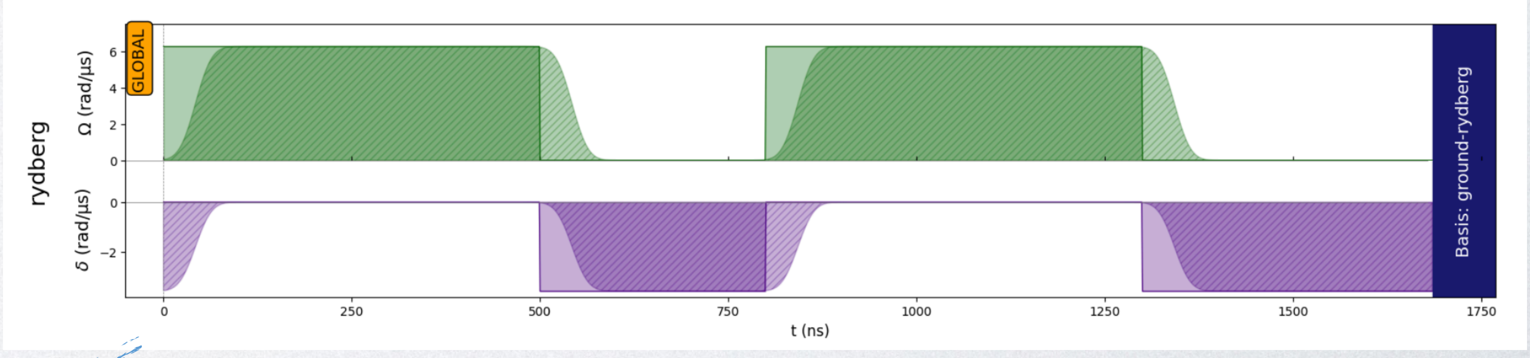

The graph of atoms is loaded on
a triangular lattice layout of traps




# COMBINATORIAL PROBLEMS ON A GRAPH




BAYESIAN
OPTIMIZATION
Step



Sequence of pulses
optimized at the previous step