Implementation, analysis and hardware-aware improvement of quantum algorithms for scientific computing.

Presented by:
Adrien Suau
LIRMM, CERFACS

Directed by:
Aida Todri-Sanial
LIRMM, CNRS

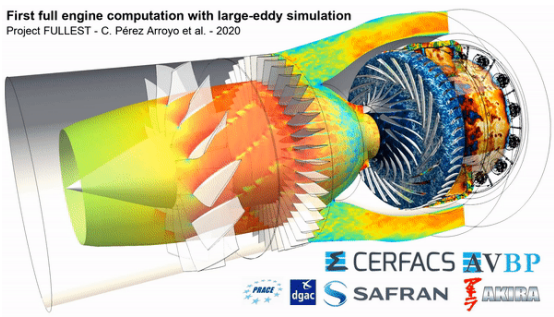
Co-directed by:
Éric Bourreau
LIRMM

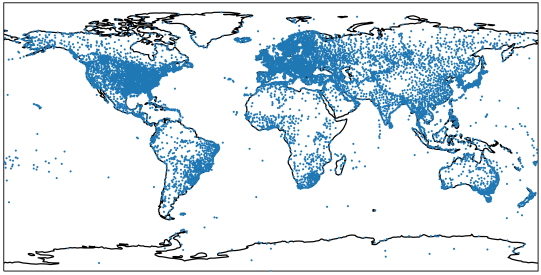
Co-directed by:
Gabriel Staffelbach
CERFACS

Industrial advisor:
Marko Rančič
TotalEnergies

Introductory examples

Scientific computing problems

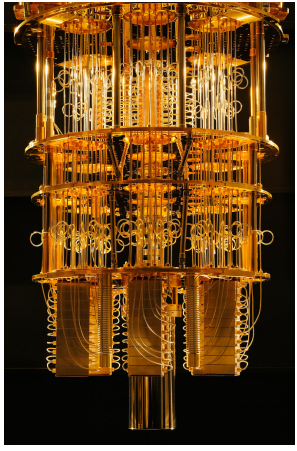




Weather stations

Quantum computing

Quantum hardware starts to appear...



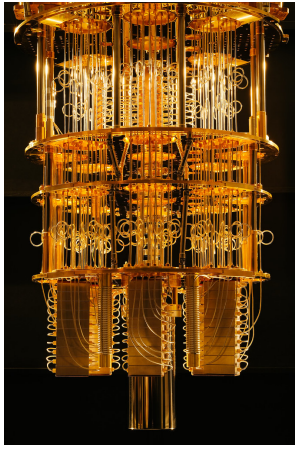
IBM quantum computer

...and a variety of quantum algorithms already exist:

- Deutsch-Jozsa
- Bernstein-Vazirani
- Shor
- Grover
- HHL

Quantum computing

Quantum hardware starts to appear...



IBM quantum computer

...and a variety of quantum algorithms already exist:

- Deutsch-Jozsa
- Bernstein-Vazirani
- Shor
- Grover
- HHL

Can we use quantum computers to solve scientific computing problems?

Scientific computing problems

Partial Differential Equations (PDEs)

Find $f : v \in E \mapsto f(v)$ a function such that

$$F\left(f, \frac{\partial f}{\partial v}, \frac{\partial^2 f}{\partial v^2}, \dots\right) = g(v)$$

with f checking appropriate *boundary* or *initial* conditions.

System of linear equations

Find $x \in \mathbb{K}^n$ such that

$$Ax = b$$

with $A \in \mathbb{K}^{2n}$ and $b \in \mathbb{K}^n$ known.

Optimisation

Find $x^* \in \mathbb{K}^n$ such that

$$x^* = \min_{x \in E} f(x)$$

with $E \subset \mathbb{K}^n$ and $f : E \to \mathbb{R}$.

Questions of the PhD

Implementation, analysis and hardware-aware improvement of quantum algorithms for scientific computing

What kind of quantum hardware would solving problem X require?

1. Choose X, the problem instance to solve
2. Implement the solver
3. Optimise the solver
4. Analyse the implementation requirements

What is currently feasible with todays quantum computing hardware?

1. Perform a state of the art of available algorithms and hardware
2. Understand the hardware requirements
3. Pick an algorithm
4. Implement the algorithm

Plan of the defense

What kind of quantum hardware would solving problem X require?

Quantum PDE solver

What are the resources requirements of such a non trivial implementation?

What is currently feasible with quantum computing and today hardware?

Solving PDE on a quantum computer

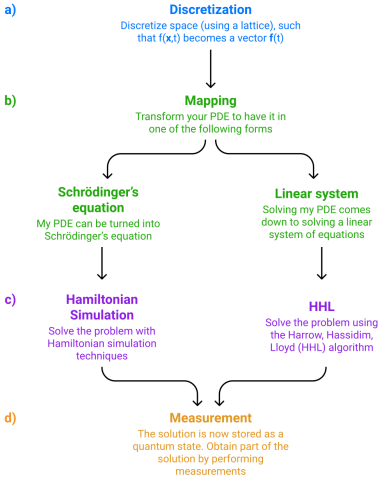


Image obtained from `https://arthurpesah.me/assets/pdf/case-study-quantum-algorithms-pde.pdf`

Schrödinger equation

A quantum state $|\psi\rangle$ evolves in time according to

$$i\frac{d}{dt}|\psi\rangle = H|\psi\rangle$$

with H the Hamiltonian of the quantum system.

Solving the Schrödinger equation

Quantum algorithms solving the Schrödinger equation are called *Hamiltonian simulation algorithms*.

Which PDE?

Wave equation

Find $\phi : ([0, 1], R^+) \to \mathbb{R}$ such that

$$\frac{\partial^2}{\partial t^2}\phi(x, t) = \frac{\partial^2}{\partial x^2}\phi(x, t)$$

with Dirichlet boundary conditions

$$\frac{\partial}{\partial x}\phi(0, t) = \frac{\partial}{\partial x}\phi(1, t) = 0.$$

A. Suau, G. Staffelbach, and H. Calandra. 2021. Practical Quantum Computing: Solving the Wave Equation Using a Quantum Approach. ACM Transactions on Quantum Computing. https://doi.org/10.1145/3430030

Which PDE?

Wave equation

Find $\phi : ([0,1], R^+) \to \mathbb{R}$ such that

$$\frac{\partial^2}{\partial t^2}\phi(x,t) = \frac{\partial^2}{\partial x^2}\phi(x,t)$$

with Dirichlet boundary conditions

$$\frac{\partial}{\partial x}\phi(0,t) = \frac{\partial}{\partial x}\phi(1,t) = 0.$$

Steps to follow

1. Translate the 1-dimensional wave equation to a Schrödinger equation.
2. Implement an efficient Hamiltonian simulation algorithm.
3. Implement the oracles.

Step 1: Translation

The translation step has already been performed in arXiv:1711.05394.

A. Suau, G. Staffelbach, and H. Calandra. 2021. Practical Quantum Computing: Solving the Wave Equation Using a Quantum Approach. ACM Transactions on Quantum Computing. https://doi.org/10.1145/3430030

Which PDE?

Wave equation

Find $\phi : ([0,1], R^+) \to \mathbb{R}$ such that

$$\frac{\partial^2}{\partial t^2}\phi(x,t) = \frac{\partial^2}{\partial x^2}\phi(x,t)$$

with Dirichlet boundary conditions

$$\frac{\partial}{\partial x}\phi(0,t) = \frac{\partial}{\partial x}\phi(1,t) = 0.$$

Steps to follow

1. Translate the 1-dimensional wave equation to a Schrödinger equation.
2. Implement an efficient Hamiltonian simulation algorithm.
3. Implement the oracles.

Step 2: Hamiltonian simulation algorithm

There are several ($\geqslant 8$) different algorithms.
We picked Trotterization-based method.

A. Suau, G. Staffelbach, and H. Calandra. 2021. Practical Quantum Computing: Solving the Wave Equation Using a Quantum Approach. ACM Transactions on Quantum Computing. https://doi.org/10.1145/3430030

Which PDE?

Wave equation

Find $\phi : ([0,1], R^+) \to \mathbb{R}$ such that

$$\frac{\partial^2}{\partial t^2}\phi(x,t) = \frac{\partial^2}{\partial x^2}\phi(x,t)$$

with Dirichlet boundary conditions

$$\frac{\partial}{\partial x}\phi(0,t) = \frac{\partial}{\partial x}\phi(1,t) = 0.$$

Steps to follow

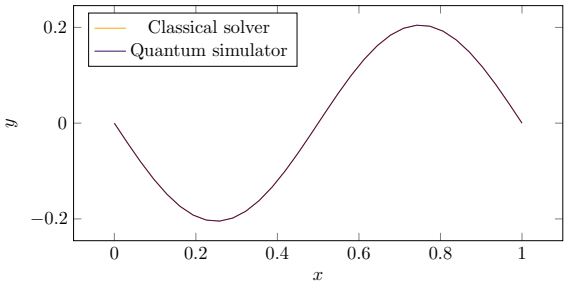
1. Translate the 1-dimensional wave equation to a Schrödinger equation.
2. Implement an efficient Hamiltonian simulation algorithm.
3. Implement the oracles.

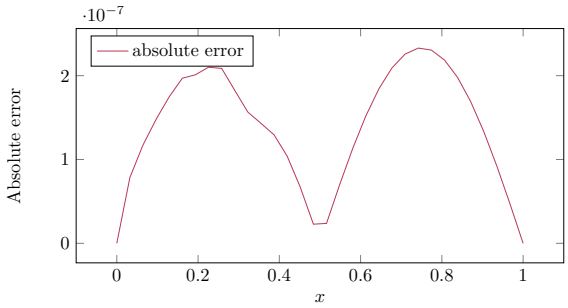
Step 3: Oracle implementation

Interface between the classical and quantum computers.

A. Suau, G. Staffelbach, and H. Calandra. 2021. Practical Quantum Computing: Solving the Wave Equation Using a Quantum Approach. ACM Transactions on Quantum Computing. https://doi.org/10.1145/3430030

Can we actually solve the 1-dimensional wave equation?

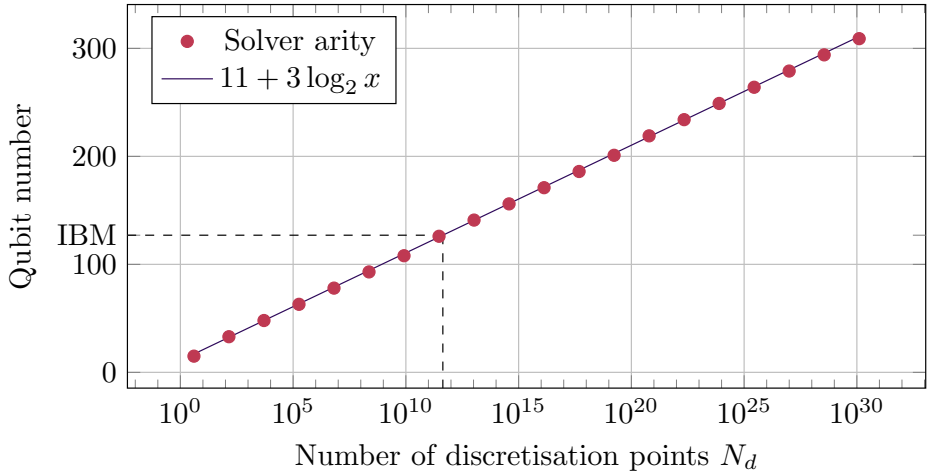




The implementation has been performed using qat and myQLM libraries from Atos and simulated with Atos simulators.

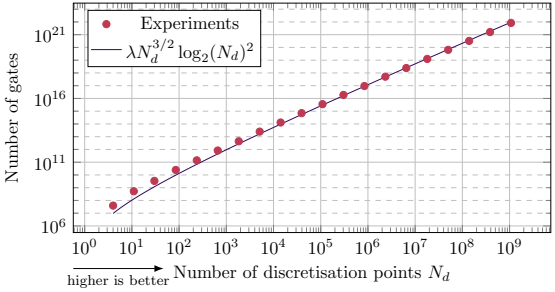
What about resources utilisation?

Qubit number

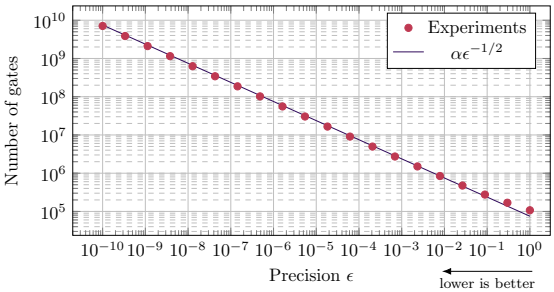


What about resources utilisation?

Gate number ($\approx$ time) scaling



(a) $\lambda = 300\,000$



(b) $\alpha = 130\,000$

Figure: Default values: Trotter-Suzuki product-formula order $k = 1$, $N = 32$ discretisation points (i.e. $n = 6$ qubits), physical time $t = 1$ and precision $\epsilon = 10^{-5}$

General remarks and knowledge obtained from the implementation

- Quantum computers **can** solve the 1-dimensional wave equation ...
- ... but it is very costly and needs *perfect* hardware.
- Manipulating and using large quantum circuits is hard
- Compilation of big quantum circuits is slow ($> 1\mathrm{h}$).
- Optimising and debugging quantum circuit implementation is a hard task.
- In particular, implementing oracles is *time-consuming* and *error-prone*.

Plan of the defense

What kind of quantum hardware would solving problem X require?

Quantum PDE solver

What are the resources requirements of such a non trivial implementation?

Quantum profiler

How to visualise and optimise quantum implementation?

What is currently feasible with quantum computing and today hardware?

qprof, the **q**uantum **prof**iler

Origin of the idea

Having the tighest constants possible for QatHS

- Perform as much optimisation as possible on the implementation
- How to optimise a quantum circuit?

qprof, the **q**uantum **prof**iler

Origin of the idea

Having the tighest constants possible for QatHS

- Perform as much optimisation as possible on the implementation
- How to optimise a quantum circuit?

Classical program optimisation

1. Benchmark the program.
2. Isolate portions of the program that takes a significant amount of resources.
3. Improve the isolated portions.
4. Come back to step 1. until desired performance is obtained.

qprof, the **q**uantum **prof**iler

Origin of the idea

Having the tighest constants possible for QatHS

- Perform as much optimisation as possible on the implementation
- How to optimise a quantum circuit?

Classical program optimisation

1. Benchmark the program.
2. Isolate portions of the program that takes a significant amount of resources.
3. Improve the isolated portions.
4. Come back to step 1. until desired performance is obtained.

qprof, the **q**uantum **prof**iler

Existing approaches in classical computing — I

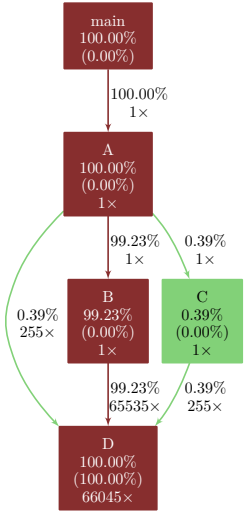
```
void D(void) {
    for(unsigned count = 0; count < 0xFFFF; count++);
}

void C(void) {
    for(unsigned count = 0; count < 0xFF; count++)
        D();
}

void B(void) {
    for(unsigned count = 0; count < 0xFFFF; count++)
        D();
}

void A(void) {
    B();
    C();
    for(unsigned count = 0; count < 0xFF; count++)
        D();
}

int main(void) {
    A();
    return 0;
}
```

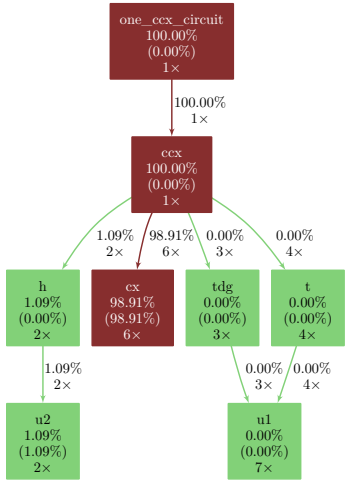


qprof, the **q**uantum **prof**iler

The qprof tool

```python
from qiskit import QuantumCircuit
from qprof import profile

############## Circuit construction ###############
circuit = QuantumCircuit(3, name="one_ccx_circuit")
circuit.ccx(0, 1, 2)
################# Hardware model ##################
gate_costs = {
    "u1": 0, "u2": 10, "u3": 30, "u": 30, "cx": 300
}
################### Profiling #####################
gprof_output = profile(
    circuit, gate_costs, "gprof", include_native_gates=True
)
with open("toffoli.qprof", "w") as f:
    f.write(gprof_output)
```



qprof, the **q**uantum **prof**iler

Important aspects of the tool

qprof is:

- Highly modular (plugins to support new frameworks and output formats).
- Cross-framework (`Qiskit`, OpenQASM 2.0, `myQLM`, `XACC`, ...).
- Very fast at profiling large quantum circuits (several orders of magnitude faster than *building* the quantum circuit)

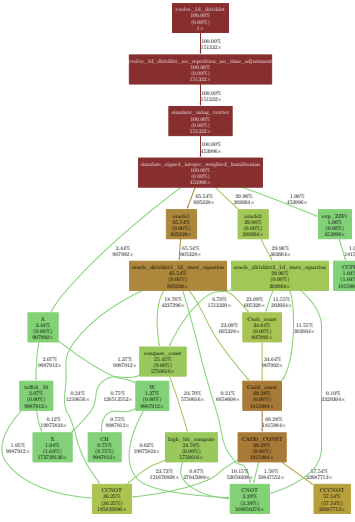
But has the following limitations:

- Only additive quantities (no ϵ).
- No gate parallelism.
- No dynamic quantum circuits.
- Not hardware-aware.

Adrien Suau, Gabriel Staffelbach, and Aida Todri-Sanial. 2022. Qprof: A gprof-Inspired Quantum Profiler. ACM Transactions on Quantum Computing. https://doi.org/10.1145/3529398

qprof, the **q**uantum **prof**iler

Results on QatHS



Insights provided by qprof

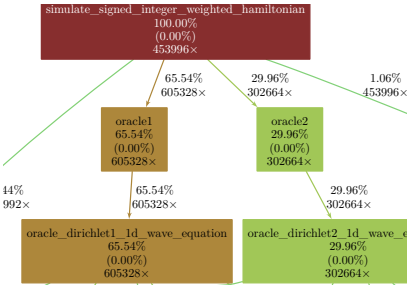
- Oracles are **costly** (95.5% of the overall gate budget).
- Addition is the most costly operation overall (nearly 70% of the overall gate budget).

Improvements performed thanks to qprof

- Replacing Draper's adder by an arithmetic-based adder (improved T-count and overall gate count).
- Hand-optimised the oracle implementations.

qprof, the **q**uantum **prof**iler

Results on QatHS



Insights provided by qprof

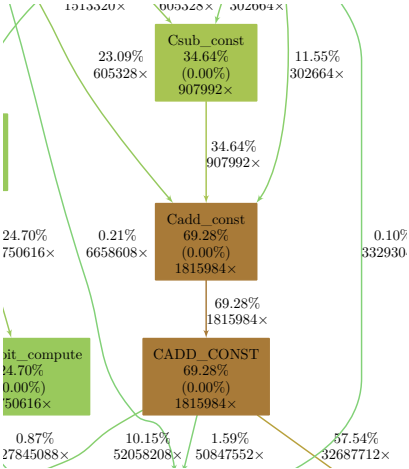
- Oracles are **costly** (95.5% of the overall gate budget).
- Addition is the most costly operation overall (nearly 70% of the overall gate budget).

Improvements performed thanks to qprof

- Replacing Draper's adder by an arithmetic-based adder (improved T-count and overall gate count).
- Hand-optimised the oracle implementations.

qprof, the **q**uantum **prof**iler

Results on QatHS



Insights provided by qprof

- Oracles are **costly** (95.5% of the overall gate budget).
- Addition is the most costly operation overall (nearly 70% of the overall gate budget).

Improvements performed thanks to qprof

- Replacing Draper's adder by an arithmetic-based adder (improved T-count and overall gate count).
- Hand-optimised the oracle implementations.

Plan of the defense

What kind of quantum hardware would solving problem X require?

Quantum PDE solver

What are the resources requirements of such a non trivial implementation?

Quantum profiler

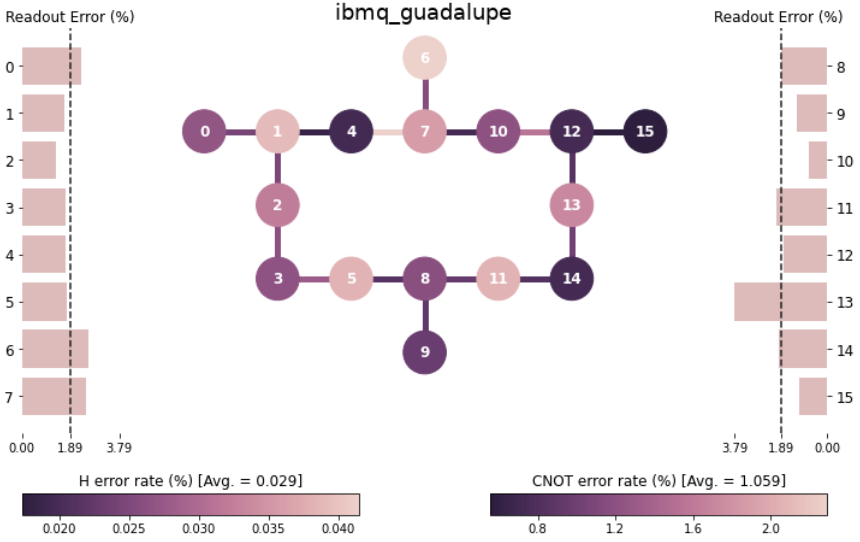
How to visualise and optimise quantum implementation?

What is currently feasible with quantum computing and today hardware?

Hardware-Aware compiler

How to use hardware calibration for better compilation?

What about quantum hardware?



Quantum “compilers”

Need to check hard conditions...

- Only use hardware-native quantum gates.
- Respect hardware topology.

...but can optimise according to:

- Total number of qubits.
- Total number of quantum gates.
- Total number of T gates.
- Total number of CX gates.
- Quantum circuit depth.

Quantum “compilers”

Need to check hard conditions...

- Only use hardware-native quantum gates.
- Respect hardware topology.

...but can optimise according to:

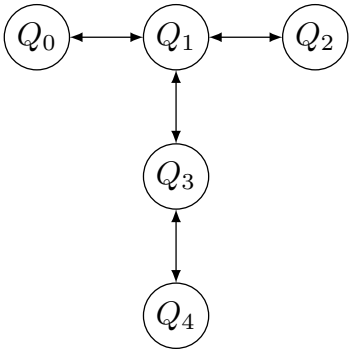
- Total number of qubits.
- Total number of quantum gates.
- Total number of T gates.
- Total number of CX gates.
- Quantum circuit depth.

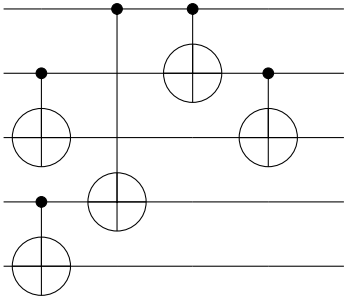
Not hardware-aware

- Quantum compiler only used connectivity.
- No adaptation with
  - T_1 and T_2 decoherence times.
  - 1-qubit gates error-rate.
  - 2-qubit gates error-rate.
  - Measurement error-rate.

A Hardware-Aware quantum compiler

Explanation of the SABRE algorithm

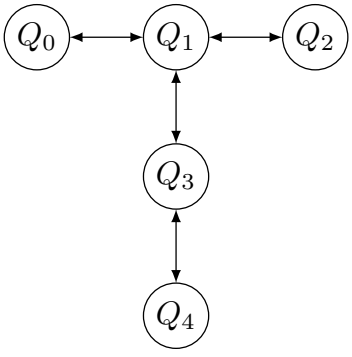


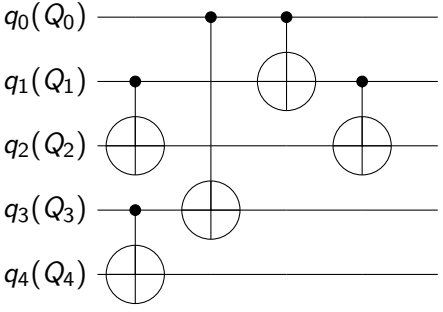


Gushu Li, et al. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. (ASPLOS '19). https://doi.org/10.1145/3297858.3304023

A Hardware-Aware quantum compiler

Explanation of the SABRE algorithm

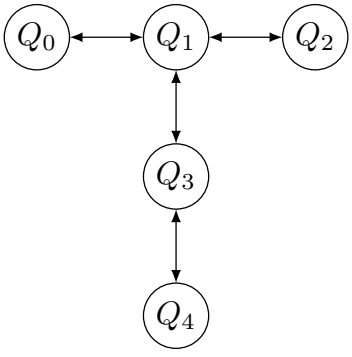


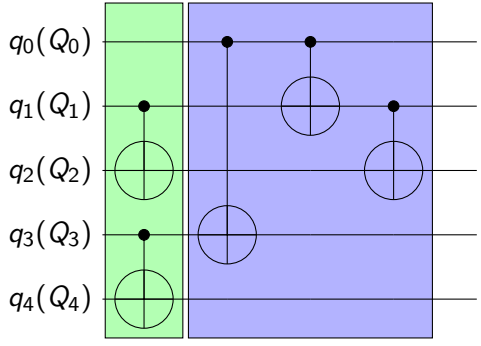


Gushu Li, et al. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. (ASPLOS '19). https://doi.org/10.1145/3297858.3304023

A Hardware-Aware quantum compiler

Explanation of the SABRE algorithm

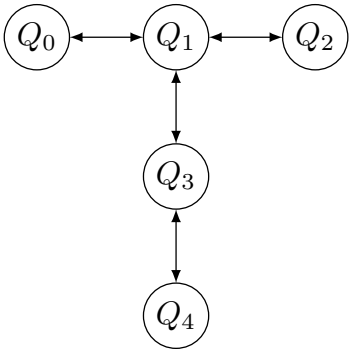


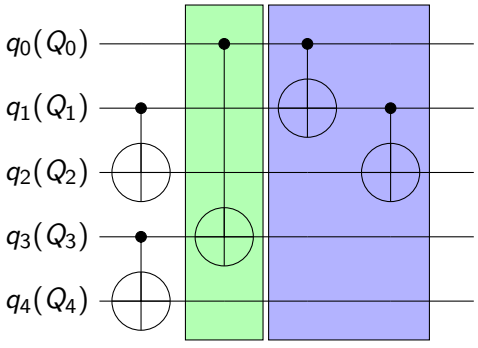


Gushu Li, et al. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. (ASPLOS '19). https://doi.org/10.1145/3297858.3304023

A Hardware-Aware quantum compiler

Explanation of the SABRE algorithm

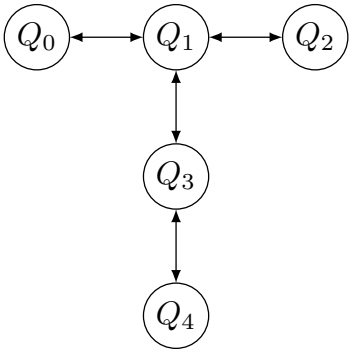


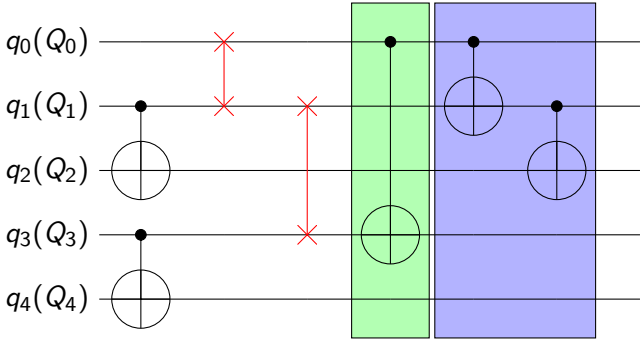


Gushu Li, et al. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. (ASPLOS '19). https://doi.org/10.1145/3297858.3304023

A Hardware-Aware quantum compiler

Explanation of the SABRE algorithm

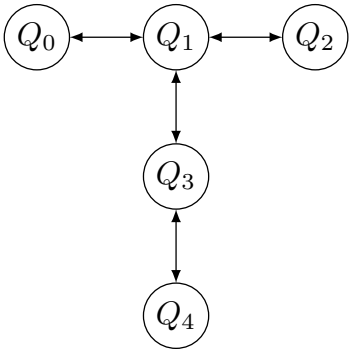


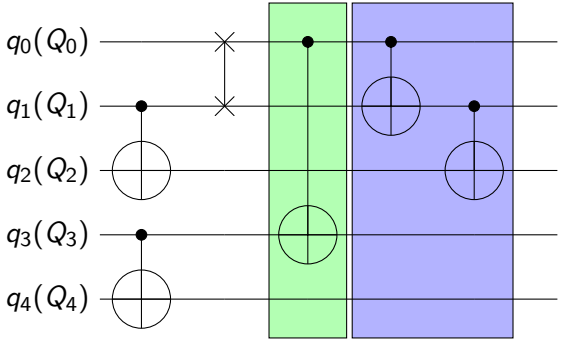


Gushu Li, et al. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. (ASPLOS '19). https://doi.org/10.1145/3297858.3304023

A Hardware-Aware quantum compiler

Explanation of the SABRE algorithm

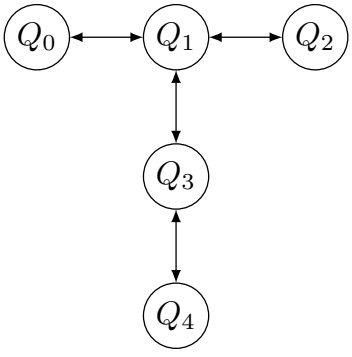


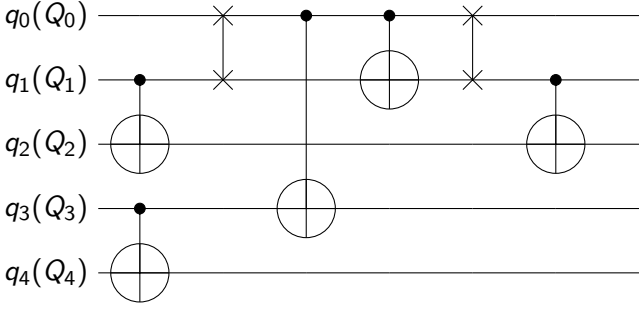


Gushu Li, et al. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. (ASPLOS '19). https://doi.org/10.1145/3297858.3304023

A Hardware-Aware quantum compiler

Explanation of the SABRE algorithm





Gushu Li, et al. 2019. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. (ASPLOS '19). https://doi.org/10.1145/3297858.3304023

A Hardware-Aware quantum compiler

Improvements to the SABRE algorithm

Improving the heuristic cost function

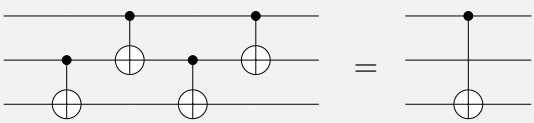
Heuristic cost function:

$$D = \alpha_1 S + \alpha_2 \mathcal{E} + \alpha_3 T$$

Include hardware-related informations into the distance D.

Adding a `Bridge` gate

The `Bridge` gate

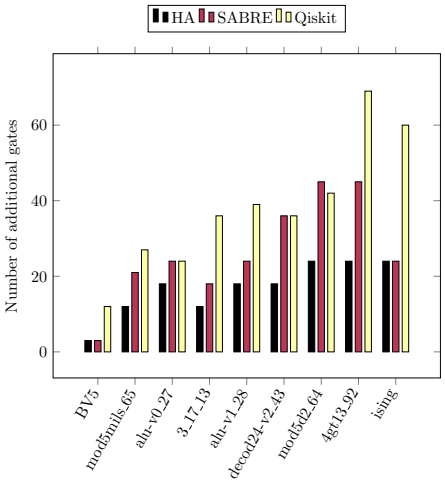


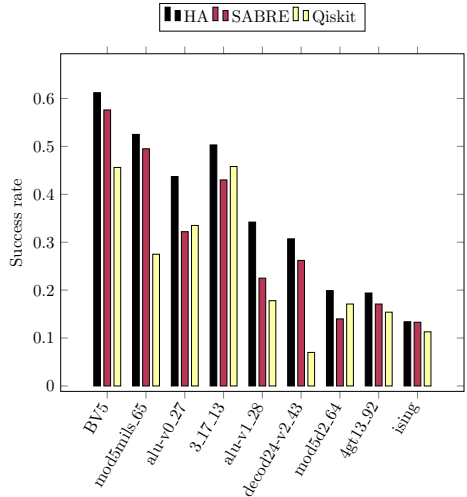
does not change the current mapping, but gives more freedom to the optimiser!

S. Niu, A. Suau, G. Staffelbach and A. Todri-Sanial, "A Hardware-Aware Heuristic for the Qubit Mapping Problem in the NISQ Era," in IEEE TQC, vol. 1, pp. 1-14, 2020, doi: 10.1109/TQE.2020.3026544.

A Hardware-Aware quantum compiler

Obtained results





Plan of the defense

What kind of quantum hardware would solving problem X require?

Quantum PDE solver

What are the resources requirements of such a non trivial implementation?

Quantum profiler

How to visualise and optimise quantum implementation?

What is currently feasible with quantum computing and today hardware?

Hardware-Aware compiler

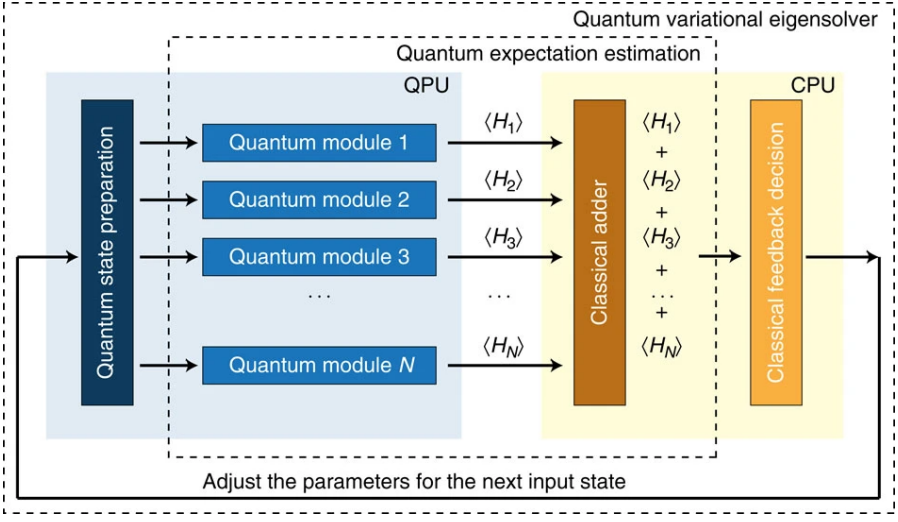
How to use hardware calibration for better compilation?

Variational Quantum Linear Solver

What kind of linear systems are we currently able to solve “quantumly”?

The Variational Quantum Linear Solver

Why variational algorithms in the first place?



The Variational Quantum Linear Solver

Important features of variational algorithms

Cost function

- Encodes the solution as a ground state.
- Might be vulnerable to Barren plateau.

Ansatz

- Can be problem-agnostic or tailored.
- Crucial for variational algorithms.

Optimisation algorithm

- Can use derivatives or not.
- Might be more adapted to a noisy cost function.

The Variational Quantum Linear Solver

Problems of interest

identity

$$A = I^{\otimes n}$$

pauli-x

$$A = \left(\bigotimes_{i=0}^{n-1} X_i \right)$$

varying_condition

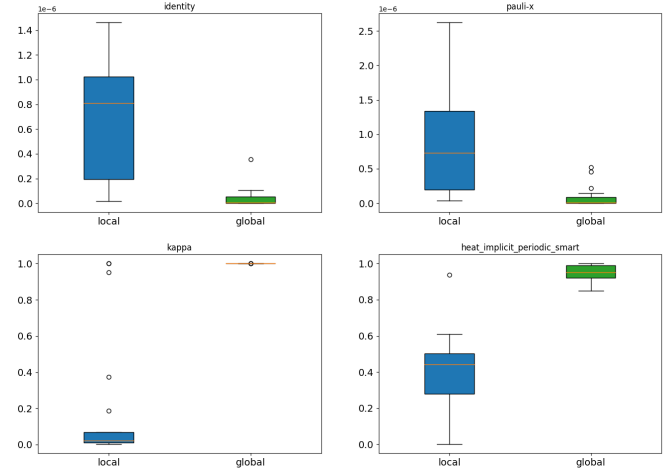
$$A(k) = \frac{k+1}{2} \bigotimes_{i=0}^{n-1} I_i + \left(1 - \frac{k+1}{2}\right) \bigotimes_{i=0}^{n-1} Z_i.$$

heat_opti

$$A = (1+2r) \left(\bigotimes_{i=0}^{n-1} I_i \right) - r\left(\mathtt{add}_1\right) - r\left(\mathtt{add}_1\right)^{\dagger}$$

The Variational Quantum Linear Solver

Global versus local cost functions

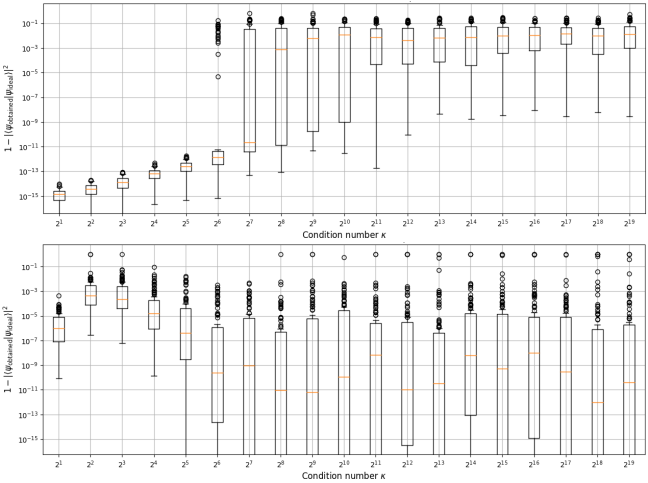


Conclusion

- Global cost function works for simple systems.
- For non-trivial systems, only the local cost function should be considered.

The Variational Quantum Linear Solver

Behaviour of different optimisers for increasing values of κ



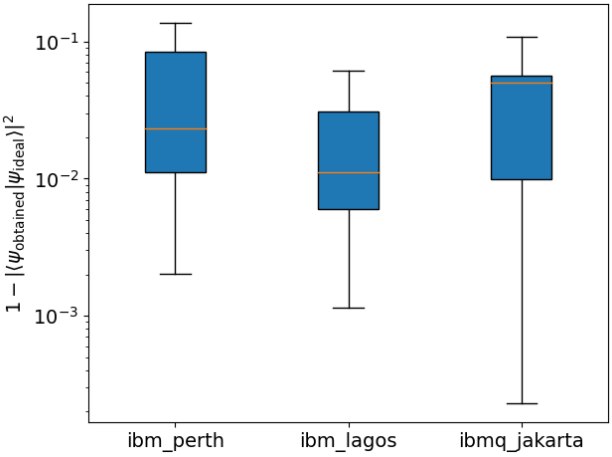
Error with increasing κ for COBYLA (up) and SPSA (down).

Conclusion

- Linear scaling for small κ.
- Plateau for higher κ.
- SPSA seems immune to plateau.

The Variational Quantum Linear Solver

What about **real** quantum hardware, `identity` linear system and COBYLA optimiser?



Conclusion

- Slightly worse than noisy simulators.
- Some runs end up close to an error of 10^{-4}!
- Most of the runs obtains a solution with an error between 0.1 and 0.01.

Plan of the defense

What kind of quantum hardware would solving problem X require?

Quantum PDE solver

What are the resources requirements of such a non trivial implementation?

Quantum profiler

How to visualise and optimise quantum implementation?

What is currently feasible with quantum computing and today hardware?

Hardware-Aware compiler

How to use hardware calibration for better compilation?

Variational Quantum Linear Solver

What kind of linear systems are we currently able to solve “quantumly”?

Conclusion

PhD contributions

Resources estimation and optimisation

- QatHS implementation and analysis.
- `qprof` profiler.

NISQ hardware compilation and usage

- Hardware-Aware quantum compiler.
- VQLS implementation and analysis.

We used quantum computing to solve PDEs or systems of linear equations, but more work needs to be done.

List of publications

Peer-reviewed

- Adrien Suau, Gabriel Staffelbach, and Aida Todri-Sanial. "Qprof: A Gprof-Inspired Quantum Profiler". In: *ACM Transactions on Quantum Computing* 4.1 (Oct. 2022). ISSN: 2643-6809. DOI: 10.1145/3529398. URL: https://doi.org/10.1145/3529398
- Adrien Suau, Gabriel Staffelbach, and Henri Calandra. "Practical Quantum Computing: Solving the Wave Equation Using a Quantum Approach". In: *ACM Transactions on Quantum Computing* 2.1 (Feb. 2021). ISSN: 2643-6809. DOI: 10.1145/3430030. arXiv: 2003.12458 [quant-ph]. URL: https://doi.org/10.1145/3430030
- S. Niu, A. Suau, et al. "A Hardware-Aware Heuristic for the Qubit Mapping Problem in the NISQ Era". In: *IEEE Transactions on Quantum Engineering* 1 (2020), pp. 1–14. DOI: 10.1109/TQE.2020.3026544

Pre-prints

- Adrien Suau, Marc Vuffray, et al. *Vector Field Visualization of Single-Qubit State Tomography*. 2022. DOI: 10.48550/ARXIV.2205.02483. URL: https://arxiv.org/abs/2205.02483
- Adrien Suau, Jon Nelson, et al. *Single-Qubit Cross Platform Comparison of Quantum Computing Hardware*. 2021. arXiv: 2108.11334 [quant-ph]
- Sanchayan Dutta, Adrien Suau, et al. *Quantum circuit design methodology for multiple linear regression*. Oct. 2020. DOI: 10.1049/iet-qtc.2020.0013. arXiv: 1811.01726 [quant-ph]. URL: https://doi.org/10.1049/iet-qtc.2020.0013

Future work and perspectives

Hierarchical quantum compiler

One major issue with current transpilers: systematic inlining!

For most of the quantum compilers available, the input quantum circuit is flattened before any other optimisation is performed.

$$e^{\lambda H} = \left[S_{2k}\left(\frac{\lambda}{n}\right) \right]^n + \mathcal{O}\left(\frac{|\lambda|^{2k+1}}{n^{2k}} \right)$$

For QatHS: $n = 453996$.

Future work and perspectives

Hierarchical quantum compiler

One major issue with current transpilers: systematic inlining!

For most of the quantum compilers available, the input quantum circuit is flattened before any other optimisation is performed.

$$e^{\lambda H} = \left[S_{2k}\left(\frac{\lambda}{n}\right)\right]^n + \mathcal{O}\left(\frac{|\lambda|^{2k+1}}{n^{2k}}\right)$$

For QatHS: $n = 453996$.

What about a "no-inline" compiler?

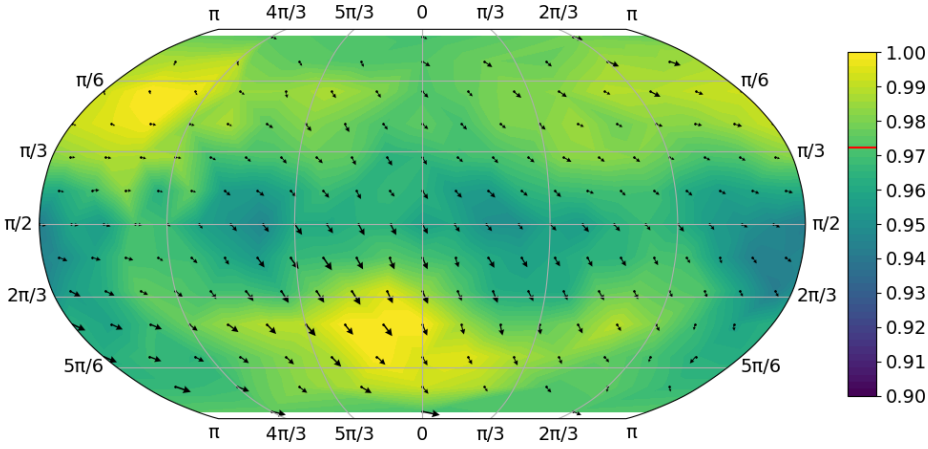
- Avoid systematic inlining
- Work on smaller circuits
- Optimise once, re-use multiple times!

Work started with 2 QAMP mentees

- Qiskit Advocate Mentorship Program (version Fall 2022)
- Adding a "no-inline" option to Qiskit transpiler

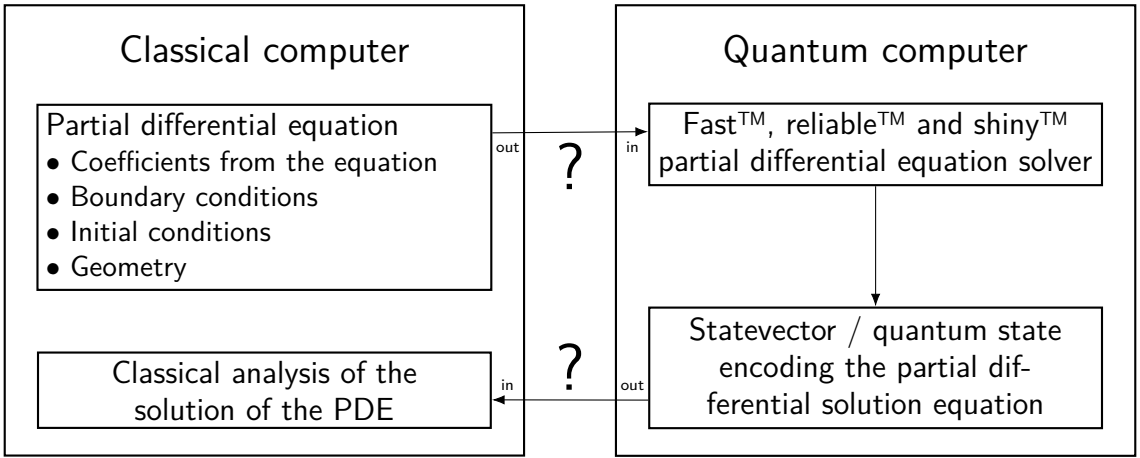
Future work and perspectives

Better understanding of quantum hardware and quantum tomography



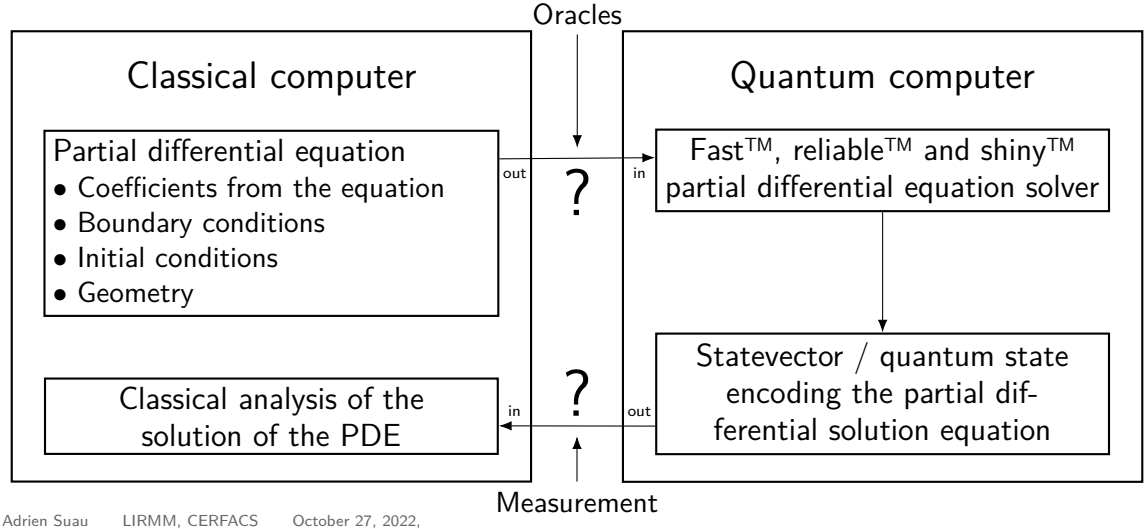
Future work and perspectives

Think about the I/O problem for PDEs



Future work and perspectives

Think about the I/O problem for PDEs



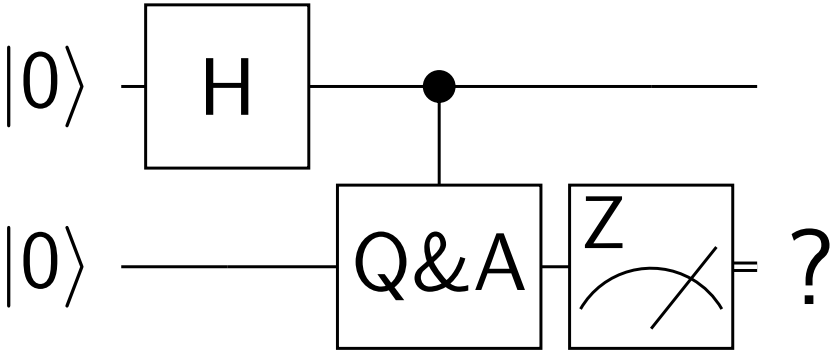
Acknowledgements

A **HUGE** thank to:

- Supervisors
  - Aida Todri-Sanial
  - Gabriel Staffelbach
  - Éric Bourreau
- Thesis committee

This work is funded by TotalEnergies

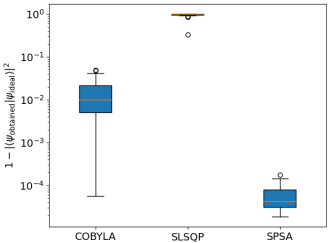
Thank you for your attention!



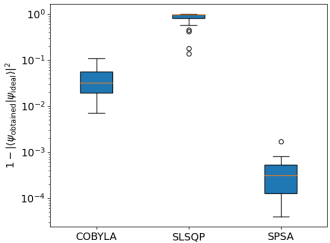
Backup slides

The Variational Quantum Linear Solver

Performance of optimisers on noisy simulators to solve the `identity` system



(a) Noisy simulator *mimicking* `ibm_lagos`



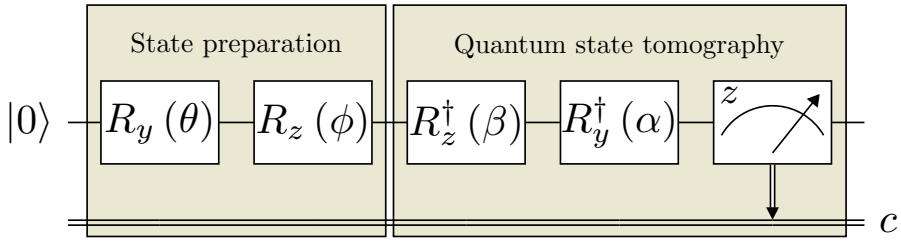
(b) Noisy simulator *mimicking* `ibmq_bogota`

Conclusion

- SLSQP (and POWELL) performs very poorly.
- COBYLA does not perform well.
- SPSA nearly always have the best result.

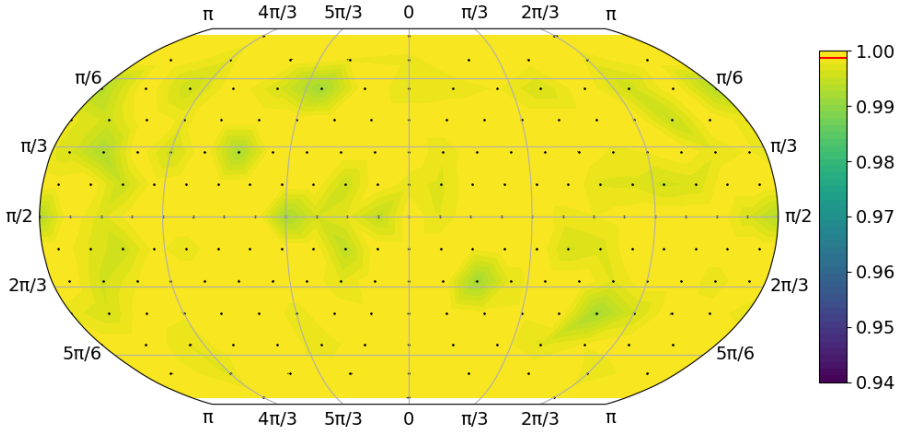
Vector field visualisation of qubit noise-map

Quantum program used to perform the experiment



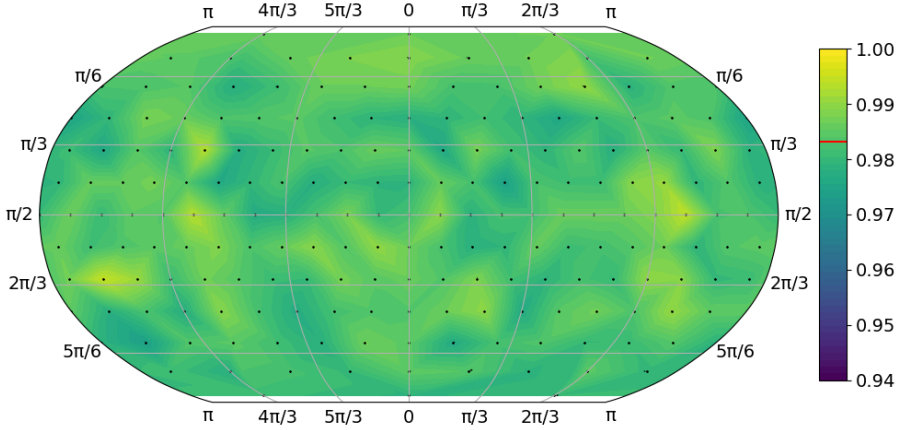
Vector field visualisation of qubit noise-map

Perfect simulator, 20 000 shots



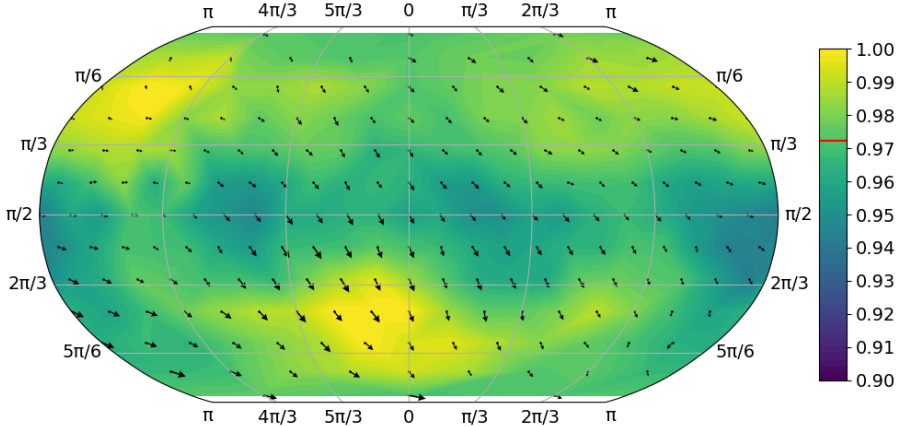
Vector field visualisation of qubit noise-map

Noisy simulator with `ibm_lagos` calibrations, 20 000 shots



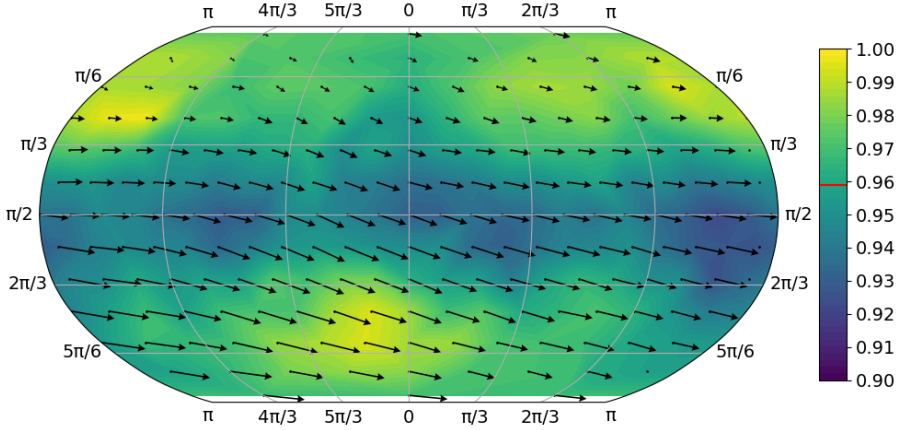
Vector field visualisation of qubit noise-map

`ibm_lagos`, 20 000 shots



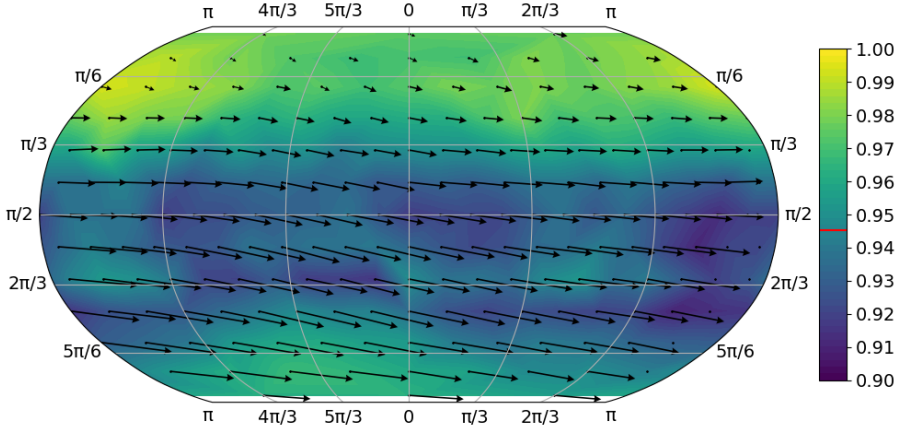
Vector field visualisation of qubit noise-map

ibm_lagos, delay of 800dt, 20 000 shots



Vector field visualisation of qubit noise-map

`ibm_lagos`, delay of $1600\mathrm{dt}$, 20 000 shots



Vector field visualisation of qubit noise-map

Absolute purity difference over 0.02 for MLE and LR reconstruction methods (probability < 0.01).