Real-time quantum error mitigation in training VQAs

Based on: arXiv:2311.05680

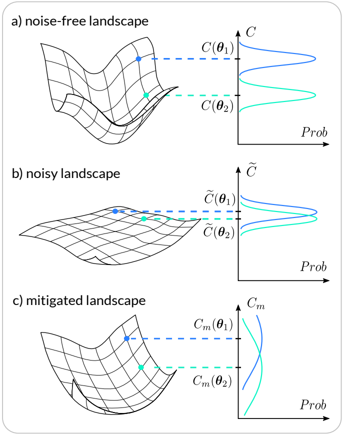
Matteo Robbiati, Alejandro Sopena, Andrea Papaluca, Stefano Carrazza

11 March 2024

Two starting points

Two starting points

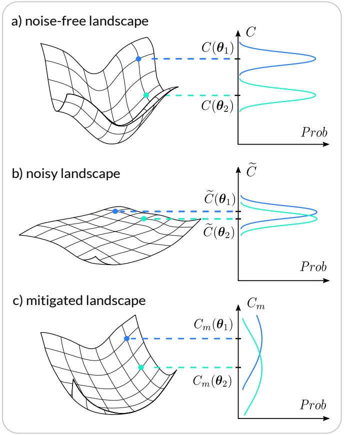
1. Noise and mitigation in QML



Credits to arXiv:2109.01051

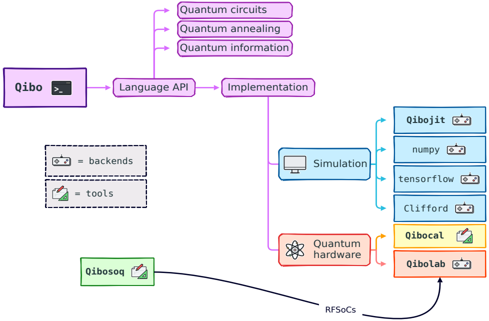
Two starting points

1. Noise and mitigation in QML



Credits to arXiv:2109.01051

2. The `Qibo` project



Full-stack framework

- </> API language with Qibo;
- quantum control with Qibolab;
- calibration with Qibocal.

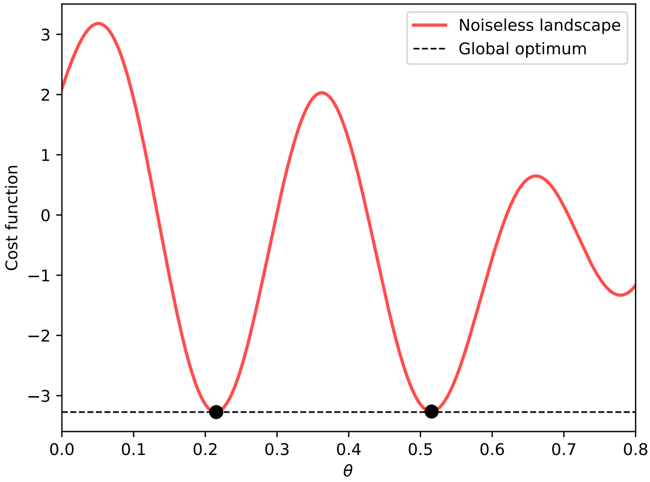
What is our goal?

What is our goal?

Use error mitigation to train on hardware, defining an algorithm which is **effective** and **computationally light**.

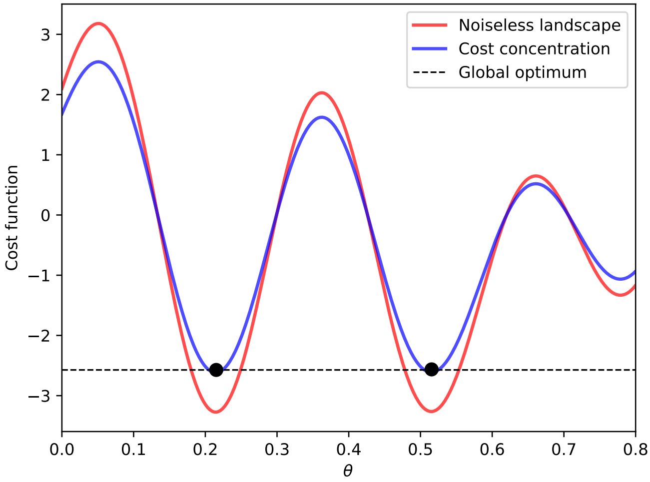
About noise and error mitigation

About noise



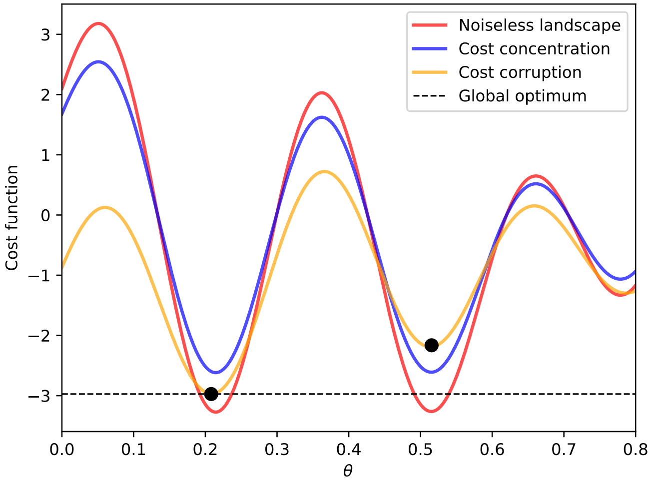
Noiseless landscape
Global optimum
Cost function
3
2
1
0
−1
−2
−3
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
θ

About noise

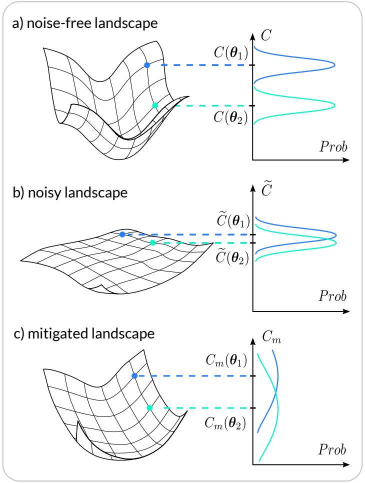


Noiseless landscape
Cost concentration
Global optimum
Cost function
3
2
1
0
−1
−2
−3
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
θ

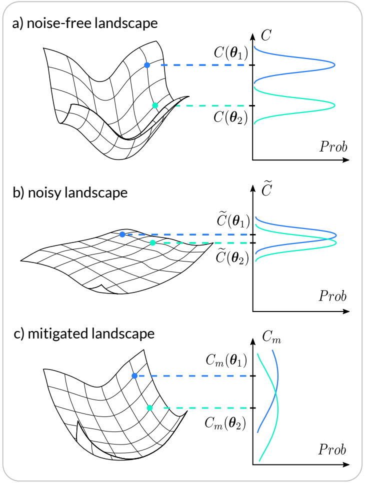
About noise



Noiseless landscape
Cost concentration
Cost corruption
Global optimum
Cost function
θ
3
2
1
0
−1
−2
−3
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

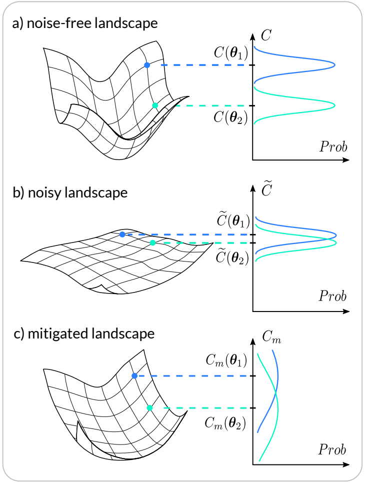


Credits to arXiv:2109.01051



Credits to arXiv:2109.01051

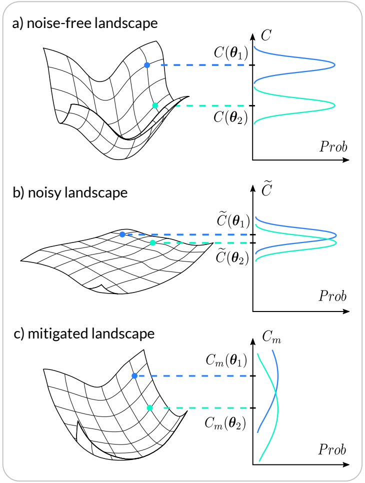
Discussion on QEM



Credits to arXiv:2109.01051

Discussion on QEM

1. Let's consider two parameters vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$;
2. thus two cost function values $C(\boldsymbol{\theta}_1)$, $C(\boldsymbol{\theta}_2)$;
3. noise and QEM affects resolvability;



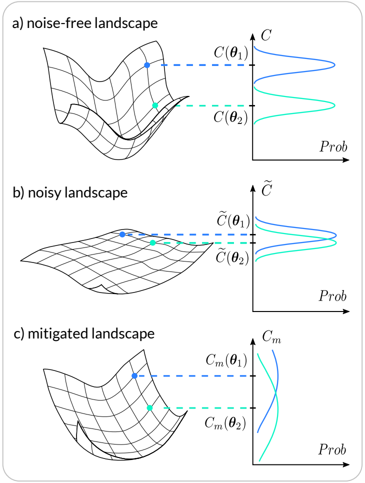
Credits to arXiv:2109.01051

Discussion on QEM

1. Let's consider two parameters vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$;
2. thus two cost function values $C(\boldsymbol{\theta}_1)$, $C(\boldsymbol{\theta}_2)$;
3. noise and QEM affects resolvability;
4. let's define a metric:

$$\chi(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \frac{N_{\text{shots}}^{\text{noisy}}}{N_{\text{shots}}^{\text{mit}}}$$

5. we are happy if $\chi \geq 1$!



Credits to arXiv:2109.01051

Discussion on QEM

1. Let's consider two parameters vectors $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$;
2. thus two cost function values $C(\boldsymbol{\theta}_1)$, $C(\boldsymbol{\theta}_2)$;
3. noise and QEM affects resolvability;
4. let's define a metric:

$$\chi(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \frac{N_{\text{shots}}^{\text{noisy}}}{N_{\text{shots}}^{\text{mit}}}$$

5. we are happy if $\chi \geq 1$!
6. for Clifford Data Regression $\chi = 1$ under Global depolarizing noise given any $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ and scaling with qubits.

Good news!

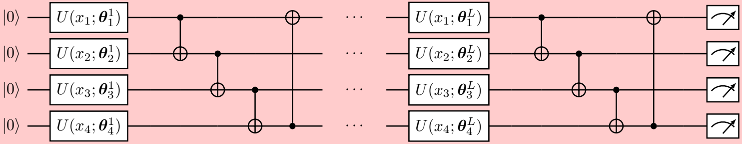
It can help with cost corruption while remaining neutral to cost concentration.

A case study

A proper target

N-dimensional fit: $y = g(\boldsymbol{x})$

We build an N-qubit parametric circuit $\mathcal{U}_{\boldsymbol{\theta}}(\boldsymbol{x})$

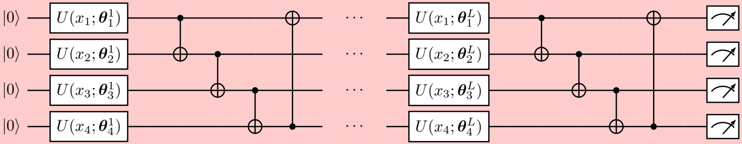


with x_j uploaded twice at layer ℓ through the uploading channel $U(x_j; \boldsymbol{\theta}_j^\ell)$.

A proper target

N-dimensional fit: $y = g(\boldsymbol{x})$

We build an N-qubit parametric circuit $\mathcal{U}_{\boldsymbol{\theta}}(\boldsymbol{x})$



with x_j uploaded twice at layer ℓ through the uploading channel $U(x_j;\boldsymbol{\theta}_j^\ell)$.

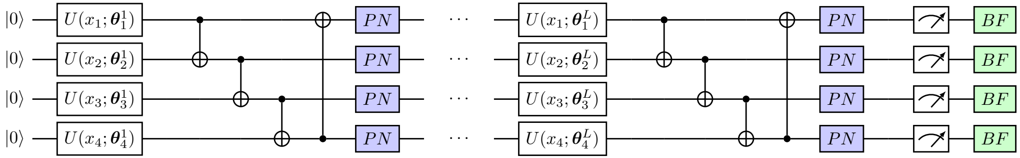
Cost function

Considering as output predictor $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \langle 0|\mathcal{U}^\dagger_{\boldsymbol{\theta}}(\boldsymbol{x})\,\sigma_z^{\otimes N}\,\mathcal{U}_{\boldsymbol{\theta}}(\boldsymbol{x})|0\rangle$, we set as cost function:

$$C_{\rm mse} = \frac{1}{N_{\rm data}} \sum_{i}^{N_{\rm data}} \left[f_{\boldsymbol{\theta}}(\boldsymbol{x}^i) - g(\boldsymbol{x}^i)\right]^2.$$

Noise model

We consider **local pauli noise** and bit-flip **readout noise** channels.

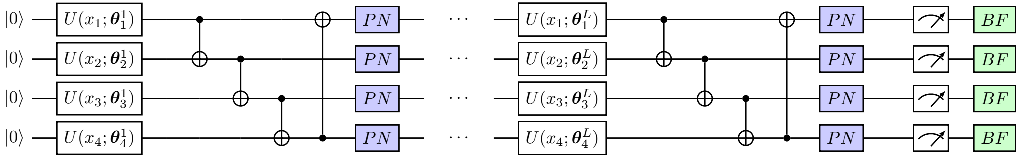


In particular:

- PN channel with probs. $-1 < q_x, q_y, q_z < 1$ on each qubit after each layer;
- symmetric readout noise $\mathcal{M}$ of single-qubit bit-flip (BF) with prob. $(1 - q_M)/2$ when measuring.

Noise model based on arXiv:2007.14384

We consider **local pauli noise** and bit-flip **readout noise** channels.



In particular:

- PN channel with probs. $-1 < q_x, q_y, q_z < 1$ on each qubit after each layer;
- symmetric readout noise $\mathcal{M}$ of single-qubit bit-flip (BF) with prob. $(1 - q_M)/2$ when measuring.

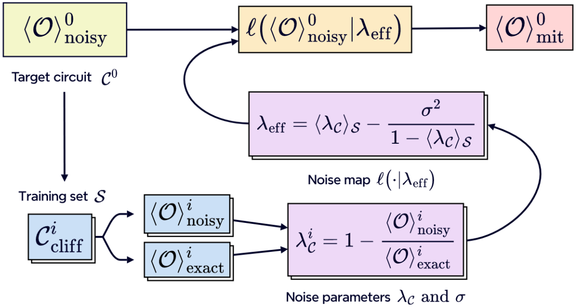
Noise effect

The effect of such a noise on our predictor is a cost concentration of the expectation values around zero:

$$|f_{\mathrm{noisy}}| < 2q_M^N q^{2l+2}\left(1 - \frac{1}{2^N}\right).$$

About error mitigation

We use the Importance Clifford Sampling (ICS) procedure to learn the noise map ℓ.



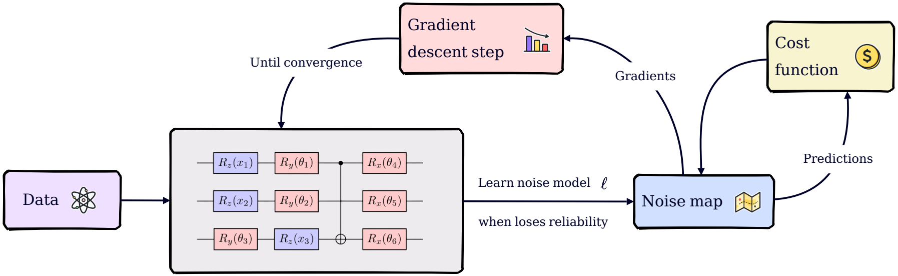
1. Sample a training set of Clifford circuits $\mathcal{S}$ on top of a target $\mathcal{C}^0$;
2. process them so that their expectation values on Pauli strings is $+1$ or -1;
3. extract mitigation parameter λ_{eff} comparing $\langle\mathcal{O}\rangle_{\text{noisy}}$ and $\langle\mathcal{O}\rangle$;
4. build a phenomenological noise map:

$$\ell(\langle O\rangle|\lambda_{\text{eff}}) = \frac{(1-\langle\lambda_{\mathcal{C}}\rangle_{\mathcal{S}})}{(1-\langle\lambda_{\mathcal{C}}\rangle_{\mathcal{S}})^2+\sigma^2}\langle O\rangle_{\text{noisy}}.$$

Real-time QEM

RTQEM pipeline

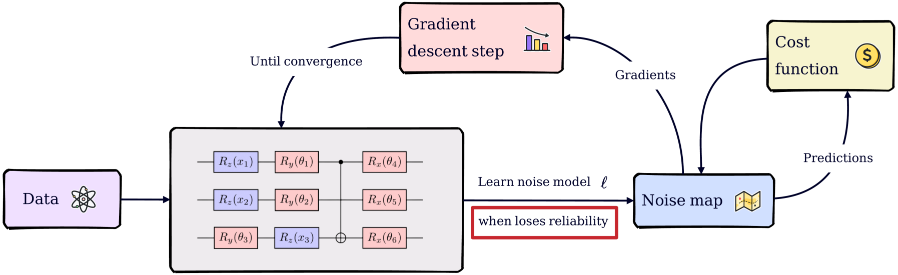
We define a Real-Time Quantum Error Mitigation (RTQEM) procedure.



1. consider a Variational Quantum Algorithm trained with gradient descent;
2. learn the noise map ℓ every time is needed over the procedure;
3. use ℓ to clean up both predictions and gradients.

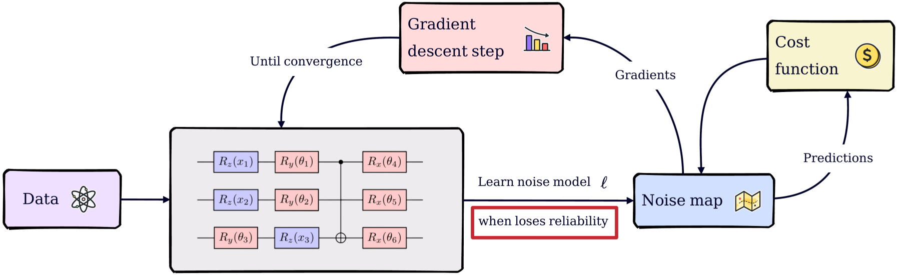
We don't need to recompute QEM at each iteration!

We define a Real-Time Quantum Error Mitigation (RTQEM) procedure.



We don't need to recompute QEM at each iteration!

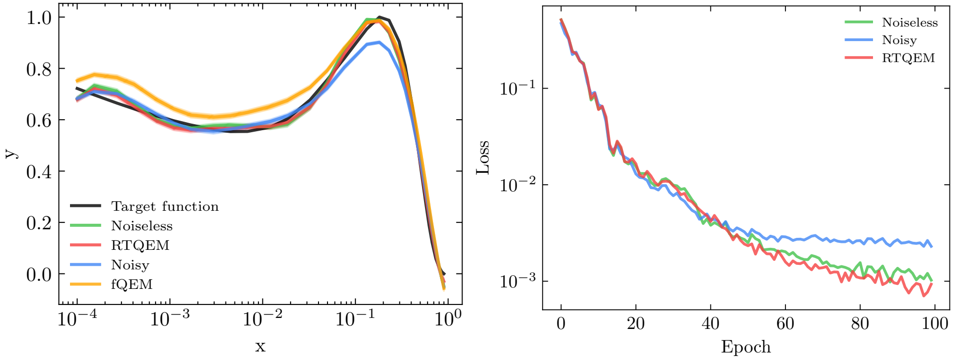
We define a Real-Time Quantum Error Mitigation (RTQEM) procedure.



- we define a metric $D(\langle z\rangle, \ell(\langle z\rangle)) = |\langle z\rangle - \ell(\langle z\rangle)|$ to quantify the distance between a well known expectation value $\langle z\rangle$ and its mitigated value.
- if D exceeds some arbitrary threshold ε, then the map ℓ is recomputed.

Static noise scenario

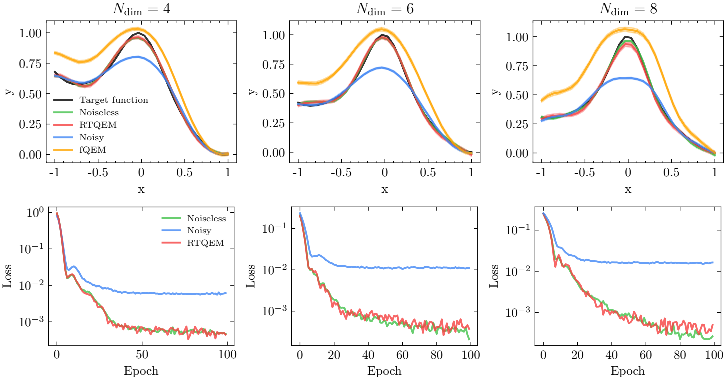
Parameter	N_{train}	N_{params}	N_{shots}	$\text{MSE}_{\text{rtqem}}$	$\text{MSE}_{\text{nomit}}$	Noise
Value	30	16	10^4	0.008	0.018	local Pauli



1. thanks to the RTQEM procedure, we reach a good minimum of the cost function;
2. the QEM is not effective is applied to a corrupted scenario (orange curve).

Dummy N-dim function: $f_{\mathrm{ndim}}(\boldsymbol{x};\boldsymbol{\beta}) = \sum_{i=1}^{N_{\mathrm{dim}}}\left[\cos\left(\beta_i x_i\right)^i + (-1)^{i-1}\beta_i x_i\right]$.

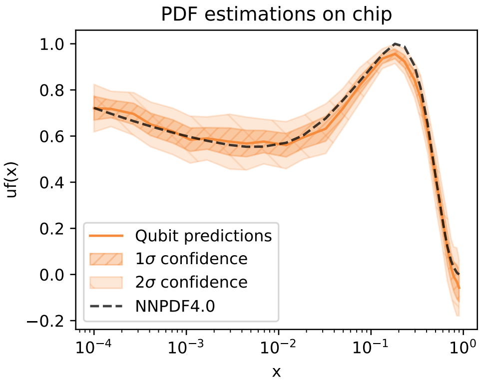
Job ID	N_{train}	N_{params}	N_{shots}	$\mathrm{MSE}_{\mathrm{rtqem}}$	$\mathrm{MSE}_{\mathrm{nomit}}$	Noise
$N_{\mathrm{dim}} = 4$	30	48	10^4	0.003	0.043	local Pauli
$N_{\mathrm{dim}} = 6$	30	72	10^4	0.002	0.083	local Pauli
$N_{\mathrm{dim}} = 8$	30	96	10^4	0.004	0.118	local Pauli

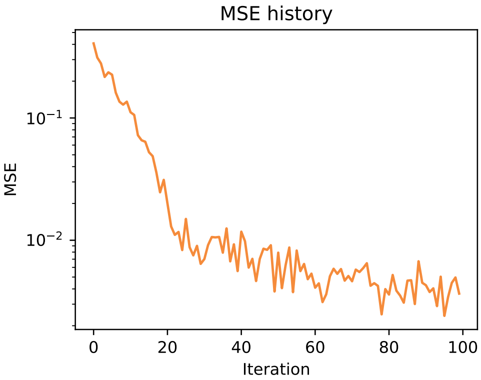


Evolving noise scenario

- we use Qibo, Qibolab and Qibocal to run a gradient descent.

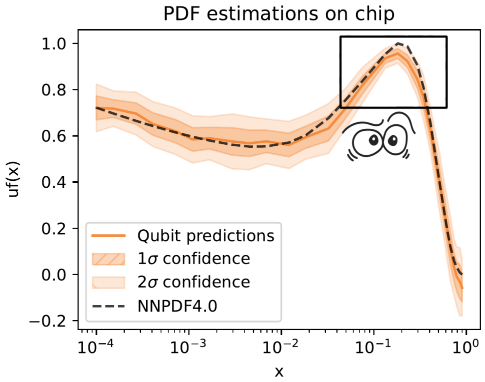
we use Qibo, Qibolab and Qibocal to run a gradient descent.

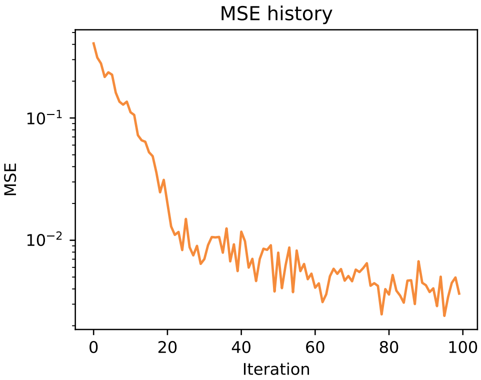




Parameter	N_{train}	N_{params}	Optimizer	N_{shots}	$\mathrm{MSE}_{\mathrm{final}}$	T_{exe}
Value	30	14	Adam	250	$3.6 \cdot 10^{-3}$	$78'$

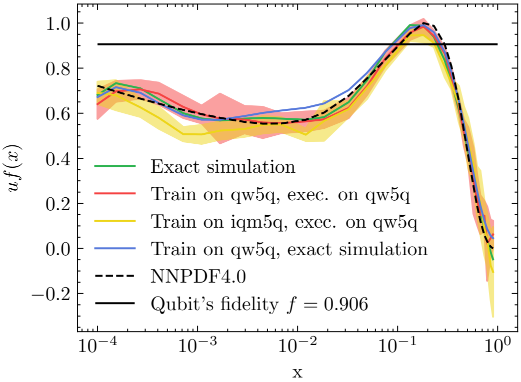
we use Qibo, Qibolab and Qibocal to run a gradient descent.





Parameter	N_{train}	N_{params}	Optimizer	N_{shots}	$\mathrm{MSE}_{\mathrm{final}}$	T_{exe}
Value	30	14	Adam	250	$3.6 \cdot 10^{-3}$	$78'$

We train on two different devices (and noises!) using the same initial conditions of the previous case.



- `qw5q` from QuantWare and controlled using Qblox instruments;
- `iqm5q` from IQM and controlled using Zurich Instruments.

Train.	**Pred.**	MSE
`qw5q`	`qw5q`	0.0013
`iqm5q`	`qw5q`	0.0037
`qw5q`	`Exact sim.`	0.0016

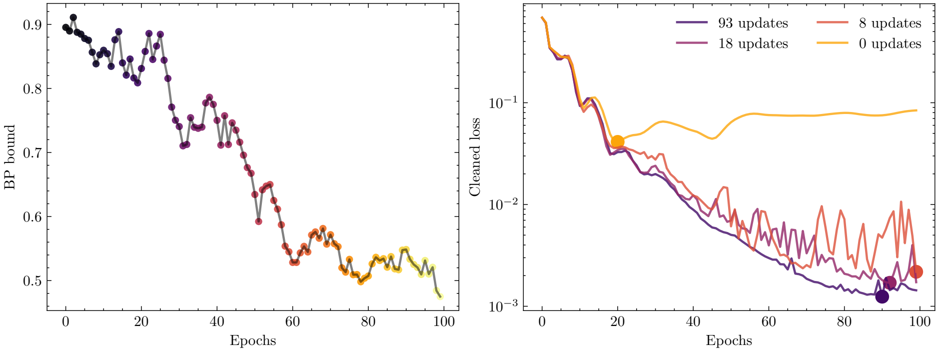
All the hardware results are obtained deploying the $\boldsymbol{\theta}_{\text{best}}$ on `qw5q`.

Simulation: RTQEM with different threshold values

We move the *PN* vector with a Random Walk. Namely, each component q_j is evolved each epoch following:

$$q_j^{(k+1)} = q_j^k + r\delta,$$

where $r \sim \{-1, +1\}$ and the step is sampled from a normal distribution $\delta \sim \mathcal{N}(0, \sigma_\delta)$.



With a limited number of updates we have a considerable advantage!

Summary and outlook

Takeway messages

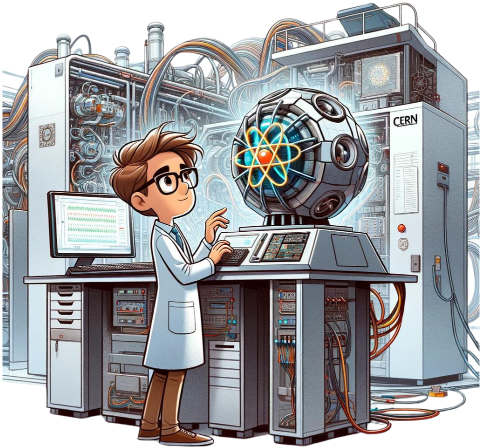
- RTQEM is **lightweight**, especially considering QML tasks!
- if the noise doesn't vary too much over time, a **few updates** of the noise map are enough.

What now?

- Can we combine **various QEM strategies**?
- Add extra features to face temporary noise fluctuations.
- Can we exploit classical accelerators to boost the process?

Some references

- arXiv:2311.05680
- https://github.com/qiboteam/rtqem.



Thank you!