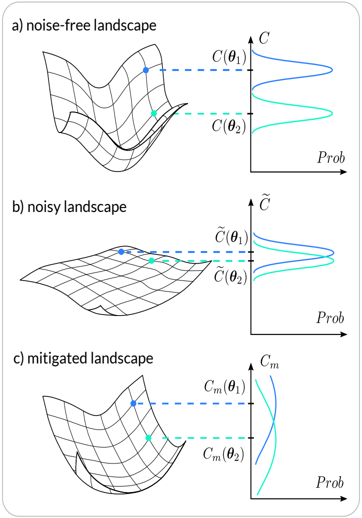
Real-time quantum error mitigation in training VQAs

Based on: arXiv:2311.05680

Matteo Robbiati, Alejandro Sopena, Andrea Papaluca, Stefano Carrazza

Prologue

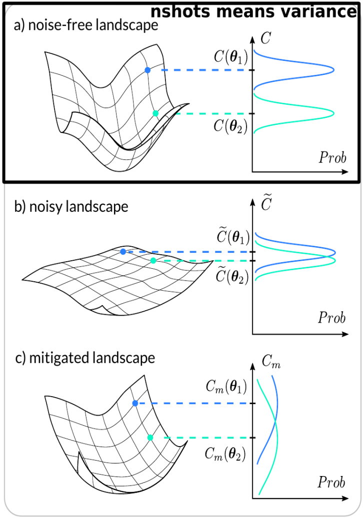
For a 360° tour `click here!`



For a 360° tour click here!

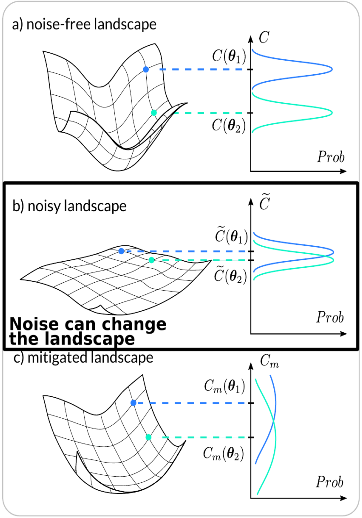
Credits to arXiv:2109.01051

For a 360° tour `click here!`

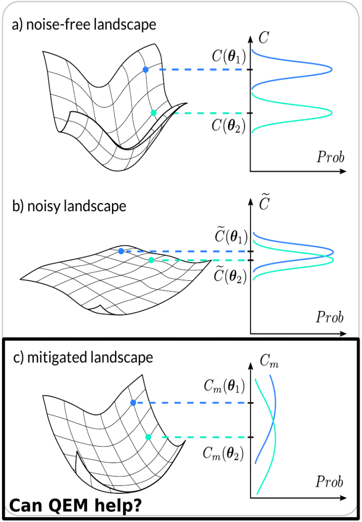


Credits to arXiv:2109.01051

For a 360° tour `click here!`



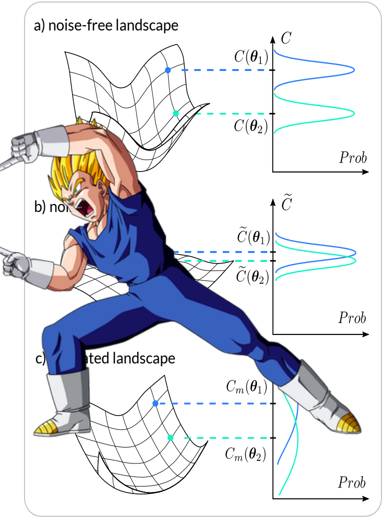
Credits to arXiv:2109.01051



For a 360° tour `click here!`

Credits to arXiv:2109.01051

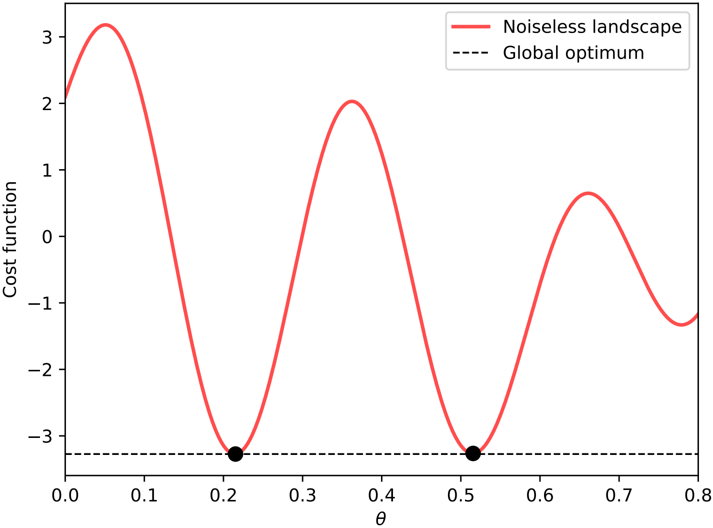
How did we start? 📘 arXiv:2109.01051



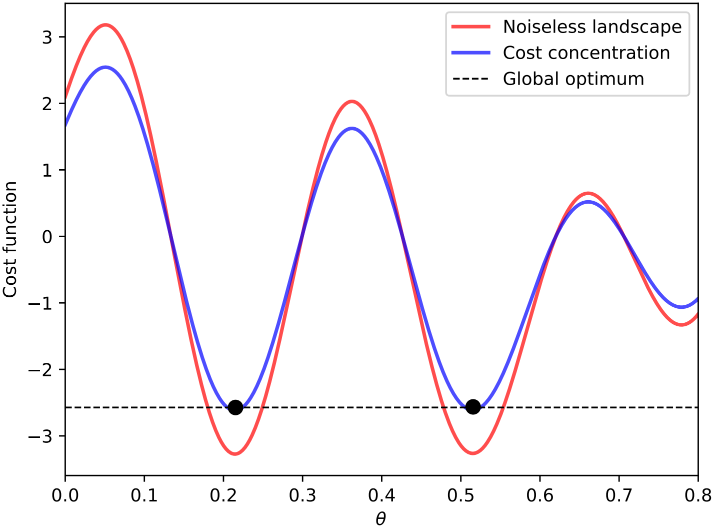
For a 360° tour `click here!`

Credits to 📘 arXiv:2109.01051

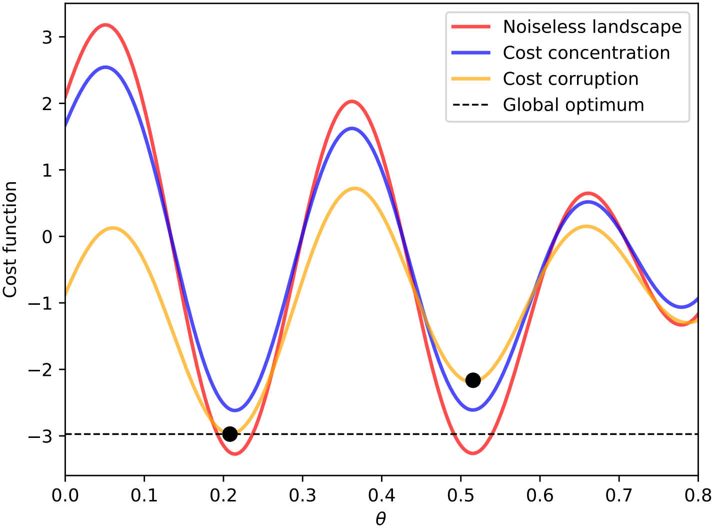
Step 1: about noise



Step 1: about noise



Step 1: about noise



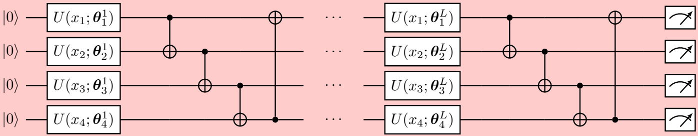
A case study

Step 2: a proper target

Step 2: a proper target

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

We build an N qubits circuit $\mathcal{U}(\boldsymbol{x};\boldsymbol{\theta})$:

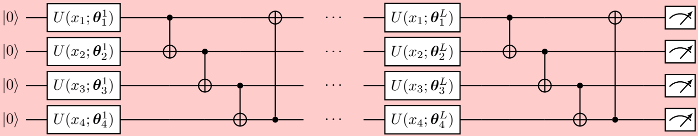


with x_j uploaded twice at layer ℓ through the uploading channel $U(x_j;\boldsymbol{\theta}_j^\ell)$.

Step 2: a proper target

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

We build an N qubits circuit $\mathcal{U}(\boldsymbol{x}; \boldsymbol{\theta})$:



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

Cost function

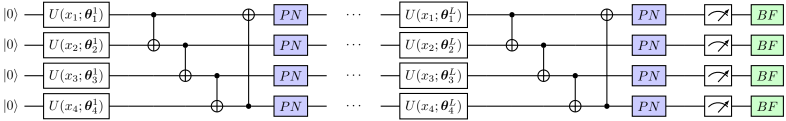
We want a cost function C which suffers of **cost corruption** and **concentration**:

$$C_{\rm mse}(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{N_{\rm data}} \sum_{i}^{N_{\rm data}} \left[f(\boldsymbol{x}^i, \boldsymbol{\theta})_{\mathcal{O}} - g(\boldsymbol{x}^i) \right]^2$$

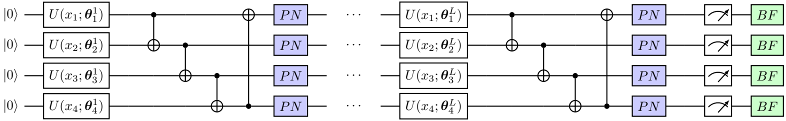
with $\quad f(\boldsymbol{x}^i, \boldsymbol{\theta})_{\mathcal{O}} = \langle \psi_0 | \mathcal{U}^\dagger(\boldsymbol{x}^i; \boldsymbol{\theta}) \, \mathcal{O} \, \mathcal{U}(\boldsymbol{x}^i; \boldsymbol{\theta}) | \psi_0 \rangle \quad$ and $\quad \mathcal{O} = \sigma_z^{\otimes N}$.

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

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



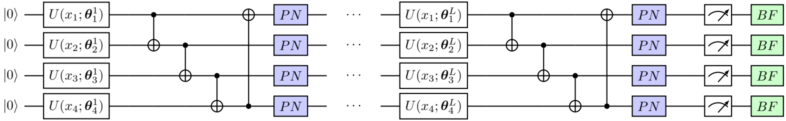
We consider local pauli noise and bit-flip readout noise channels.



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. $(1 - q_M)/2$.

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

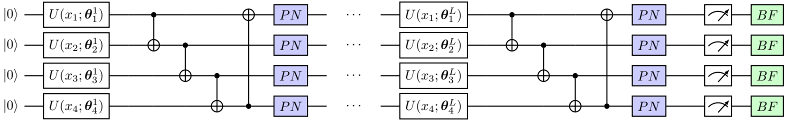


In particular:

- 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. $(1 - q_M)/2$.

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

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



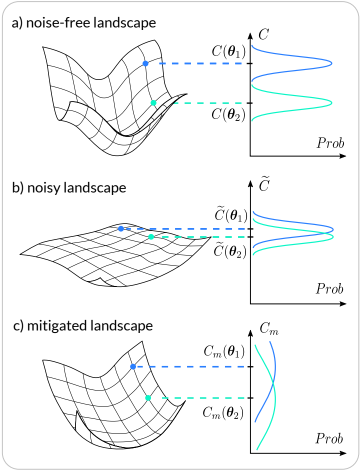
In particular:

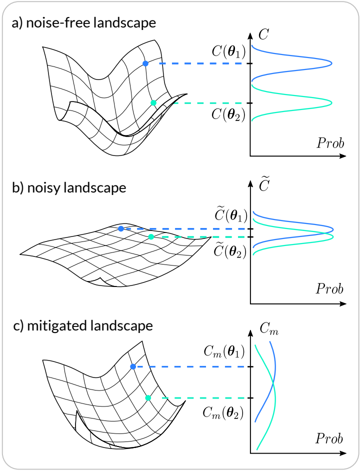
- *PN* channel with probs. $-1 < q_x, q_y, q_z < 1$ on each qubit after each layer;
- symmetric readout noise $\mathcal{M}$ of single-qubit bit-flip (BF) with prob. $(1 - q_M)/2$.

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

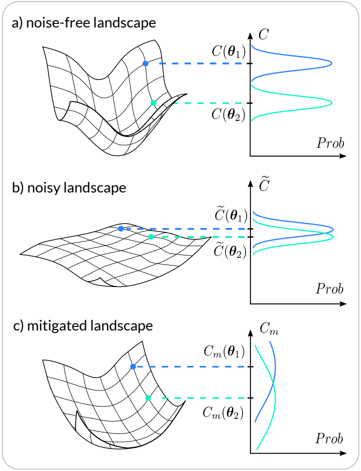
$$|f_{\mathrm{noisy}}| < 2q_M^N q^{2l+2}\left(1 - \frac{1}{2^N}\right)$$

Can quantum error mitigation help? arXiv:2109.01051

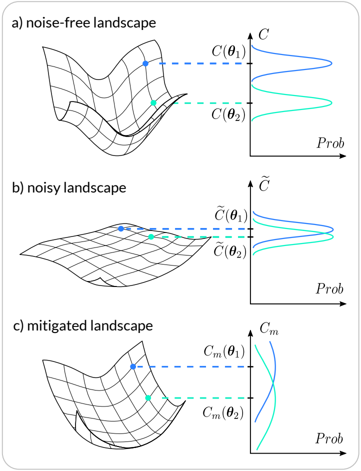




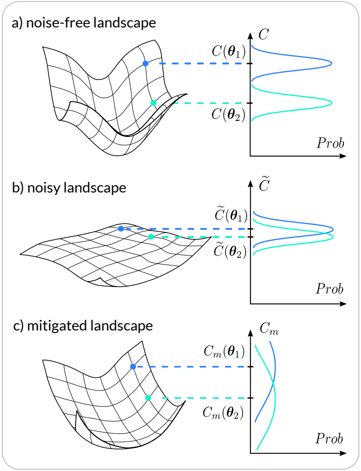
1. Let's consider $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$;



1. Let's consider $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$;
2. thus two cost $C(\boldsymbol{\theta}_1)$, $C(\boldsymbol{\theta}_2)$;

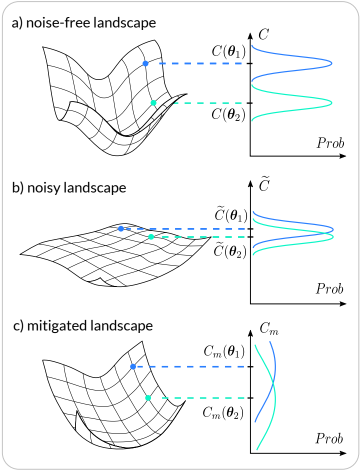


1. Let's consider $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$;
2. thus two cost $C(\boldsymbol{\theta}_1)$, $C(\boldsymbol{\theta}_2)$;
3. noise and QEM affects resolvability;



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

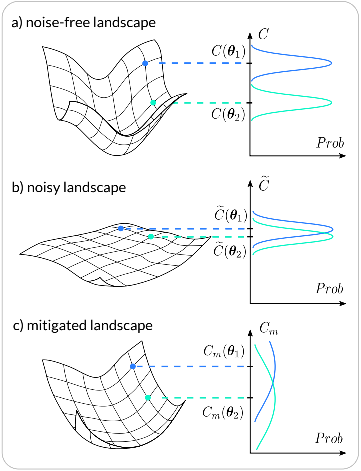
$$\chi(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \frac{N_{\text{shots}}^{\text{noisy}}}{N_{\text{shots}}^{\text{mit}}}$$



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

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

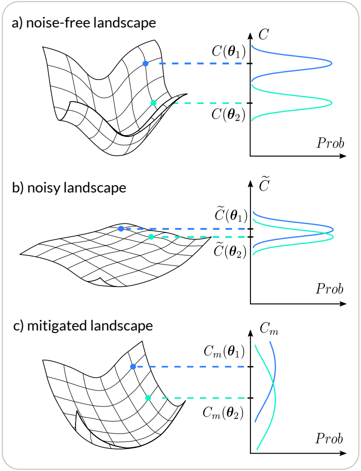
5. we like it if $\chi \geq 1$!



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

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

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



1. Let's consider $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$;
2. thus two cost $C(\boldsymbol{\theta}_1)$, $C(\boldsymbol{\theta}_2)$;
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 like it if $\chi \geq 1$!
6. for Clifford Data Regression $\chi = 1$ under Global depolarizing noise given any $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$.

Good news!

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

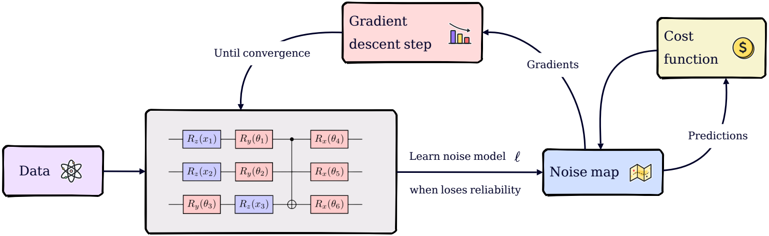
can we use it efficiently in VQAs?

RTQEM pipeline

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

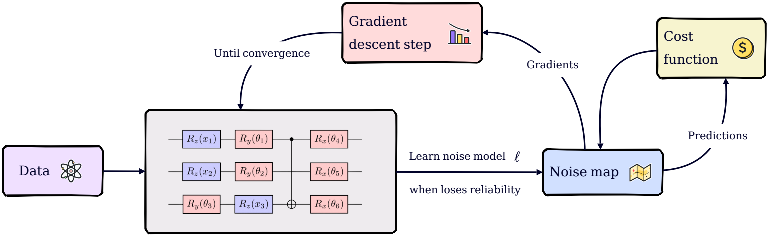
RTQEM pipeline

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



RTQEM pipeline

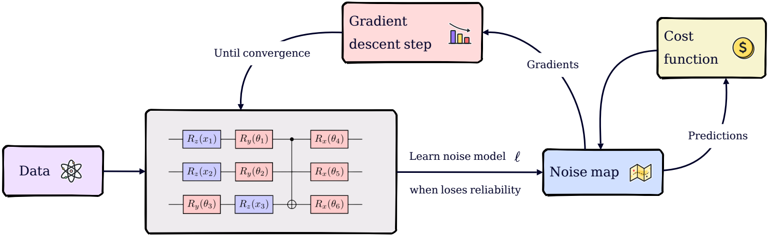
We define a Real-Time Quantum Error Mitigation (RTQEM) procedure.



1. consider a Variational Quantum Algorithm trained with gradient descent;

RTQEM pipeline

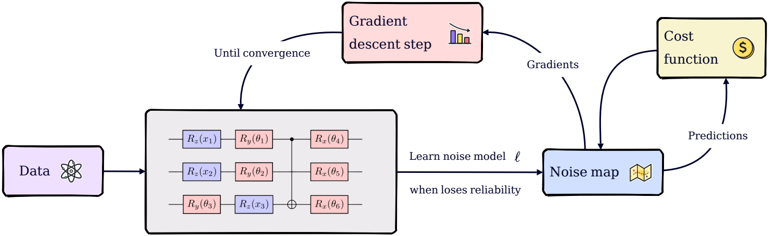
We define a Real-Time Quantum Error Mitigation (RTQEM) procedure.



1. consider a Variational Quantum Algorithm trained with gradient descent;
2. learn the noise map ℓ every time is needed over the procedure;

RTQEM pipeline

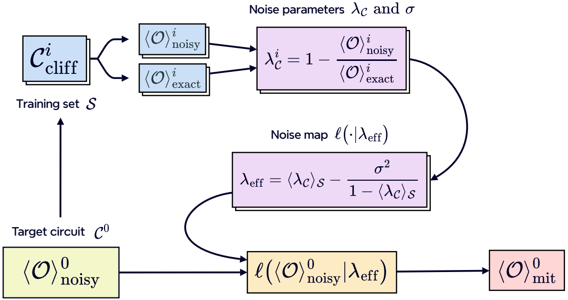
We define a Real-Time Quantum Error Mitigation (RTQEM) procedure.



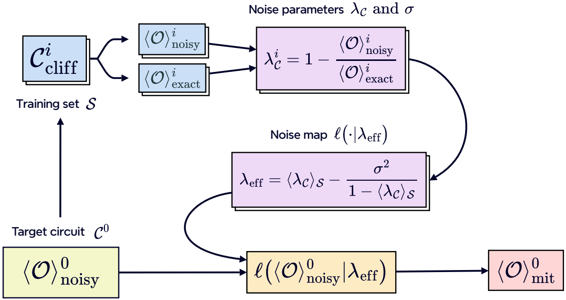
1. consider a Variational Quantum Algorithm trained with gradient descent;
2. learn the noise map ℓ every time is needed over the procedure;
3. use ℓ to clean up both predictions and gradients.

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

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

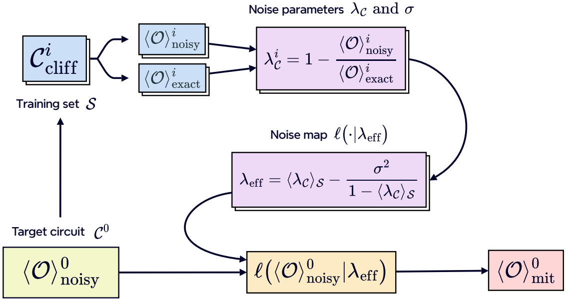


We use the Importance Clifford Sampling (ICS) procedure to learn the noise map ℓ.



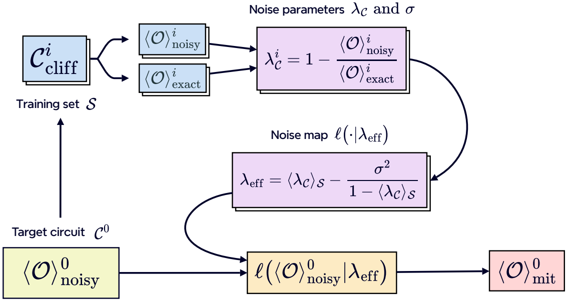
1. sample a training set of Clifford circuits $\mathcal{S}$ on top of a target $\mathcal{C}^0$;

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



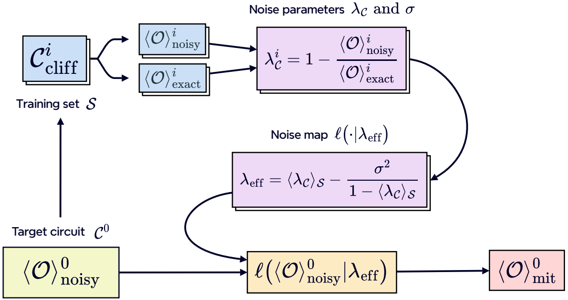
1. sample a training set of Clifford circuits $\mathcal{S}$ on top of a target $\mathcal{C}^0$;
2. process them so that their expectation values on Pauli strings is $+1$ or -1;

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 $\langle\hat{\mathcal{O}}\rangle_{\text{noisy}}$ and $\langle\hat{\mathcal{O}}\rangle$;

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



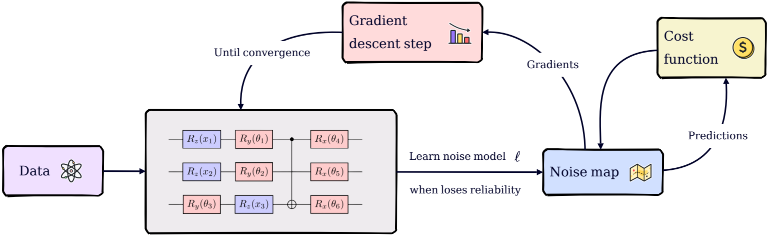
1. sample a training set of Clifford circuits $\mathcal{S}$ on top of a target $\mathcal{C}^0$;
2. process them so that their expectation values on Pauli strings is $+1$ or -1;
3. extract mitigation parameter λ_{eff} comparing $\langle\hat{\mathcal{O}}\rangle_{\text{noisy}}$ and $\langle\hat{\mathcal{O}}\rangle$;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

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

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

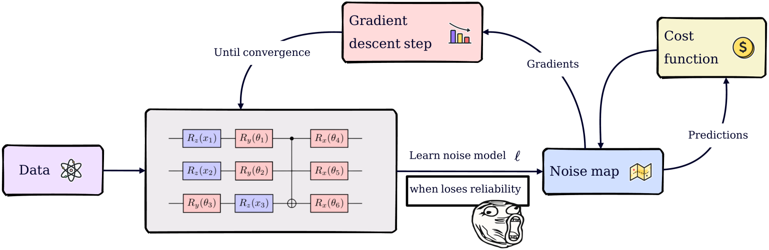
We define a Real-Time Quantum Error Mitigation (RTQEM) procedure.

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



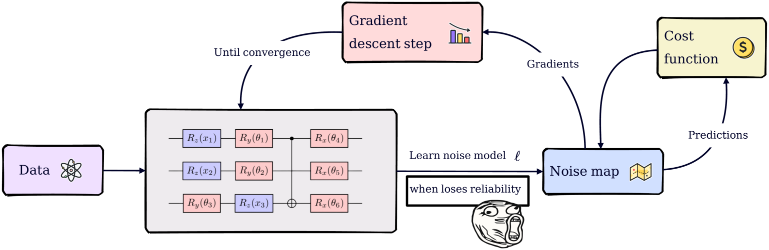
We don't need to recompute QEM at each iteration!

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



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

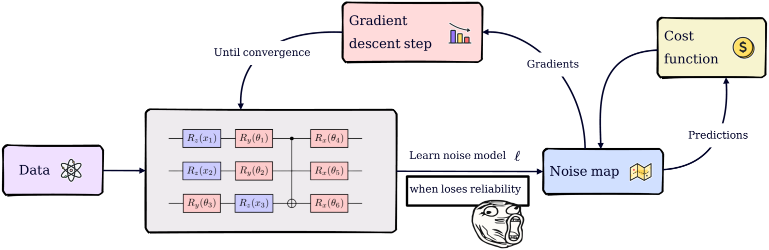
We define a Real-Time Quantum Error Mitigation (RTQEM) procedure.



✎ we defined a metric $D(z, \ell(z)) = |z - \ell(z)|$ to quantify the distance between a well known expval z and its mitigated value.

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

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



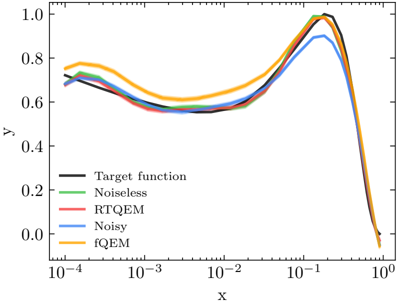
we defined a metric $D(z, \ell(z)) = |z - \ell(z)|$ to quantify the distance between a well known expval z and its mitigated value.

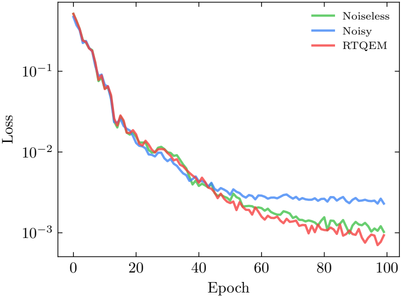
if D exceeds some arbitrary threshold ε, then the map ℓ is recomputed.

Static noise scenario

One dimensional HEP target: the u-quark PDF

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



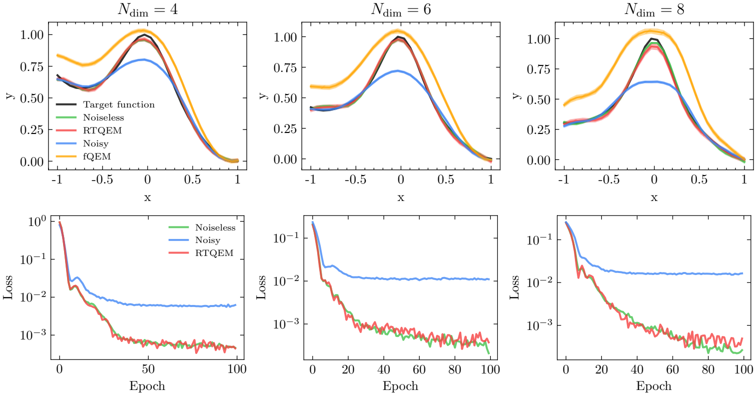


1. thanks to the RTQEM procedure, we reach a good minimum of the cost function;
2. the QEM is not effective is applied to a corrupted scenario (orange curve).

Multidimensional target

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

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

RTQEM on a superconducting qubit

We move the *PN* vector with a Random Walk-like procedure. Namely, each component q_j is evolved from epoch k to epoch $k+1$ as

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

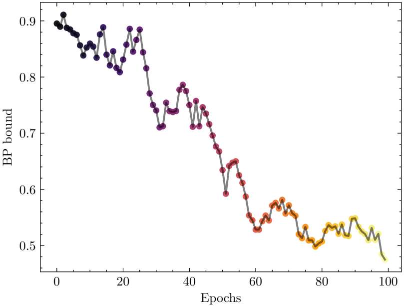
where $r \sim \{-1, +1\}$ and the step is sampled from a normal distribution $\delta \sim \mathcal{N}(0, \sigma_\delta)$.

RTQEM on a superconducting qubit

We move the *PN* vector with a Random Walk-like procedure. Namely, each component q_j is evolved from epoch k to epoch $k+1$ as

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

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

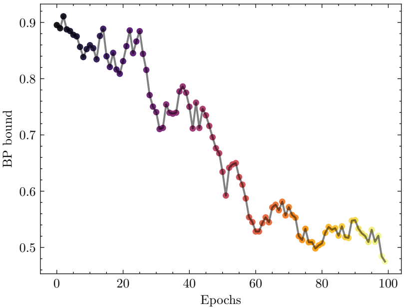


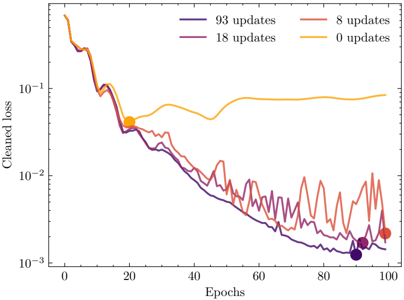
RTQEM on a superconducting qubit

We move the *PN* vector with a Random Walk-like procedure. Namely, each component q_j is evolved from epoch k to epoch $k+1$ as

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

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



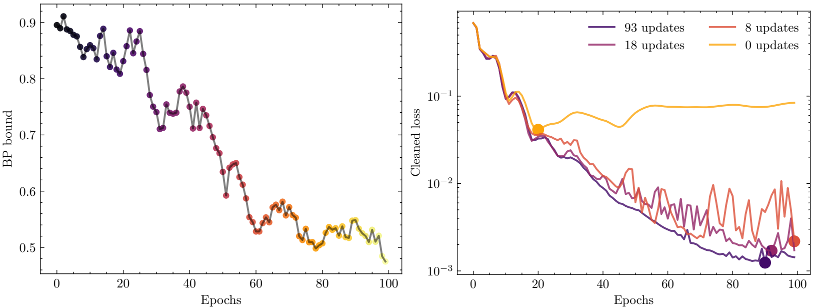


RTQEM on a superconducting qubit

We move the *PN* vector with a Random Walk-like procedure. Namely, each component q_j is evolved from epoch k to epoch $k+1$ as

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

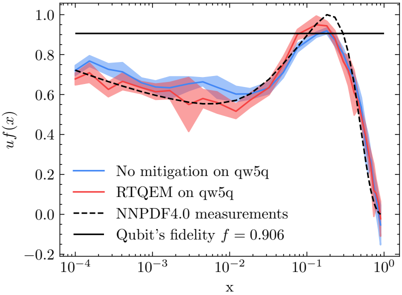
where $r \sim \{-1, +1\}$ and the step is sampled from a normal distribution $\delta \sim \mathcal{N}(0, \sigma_\delta)$.

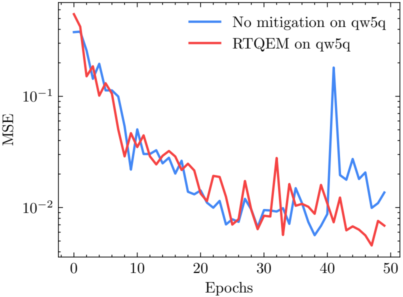


With a limited number of updates we have a considerable advantage!

RTQEM on a superconducting qubit

Parameter	N_{train}	N_{params}	N_{shots}	$\mathrm{MSE}_{\mathrm{rtqem}}$	$\mathrm{MSE}_{\mathrm{nomit}}$	Noise
Value	15	16	500	0.0042	0.0055	real noise

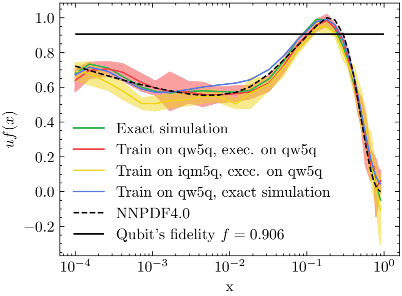




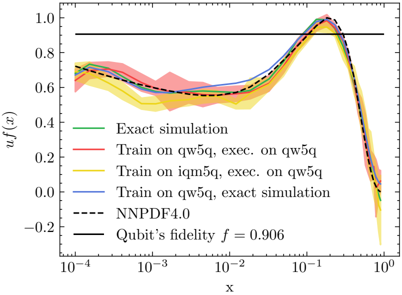
RTQEM allows exceeding the natural bound imposed by noise.

We perform a longer training on two different devices (and noises!) using the same initial conditions of the previous slide but $N_{\mathrm{epochs}} = 100$.

We perform a longer training on two different devices (and noises!) using the same initial conditions of the previous slide but $N_{\text{epochs}} = 100$.

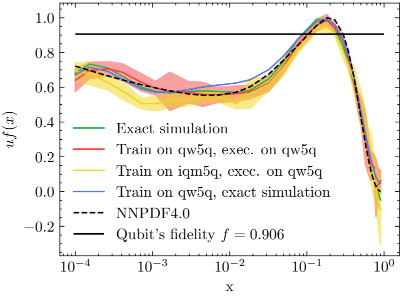


We perform a longer training on two different devices (and noises!) using the same initial conditions of the previous slide but $N_{\text{epochs}} = 100$.



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

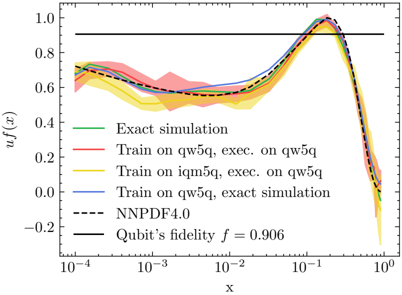
We perform a longer training on two different devices (and noises!) using the same initial conditions of the previous slide but $N_{\text{epochs}} = 100$.



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

Train.	Epochs	Pred.	Config.	MSE
`qw5q`	50	`qw5q`	noisy	0.0055
`qw5q`	50	`qw5q`	RTQEM	0.0042
`qw5q`	100	`qw5q`	RTQEM	0.0013
`iqm5q`	100	`qw5q`	RTQEM	0.0037
`qw5q`	100	`sim`	RTQEM	0.0016

We perform a longer training on two different devices (and noises!) using the same initial conditions of the previous slide but $N_{\text{epochs}} = 100$.



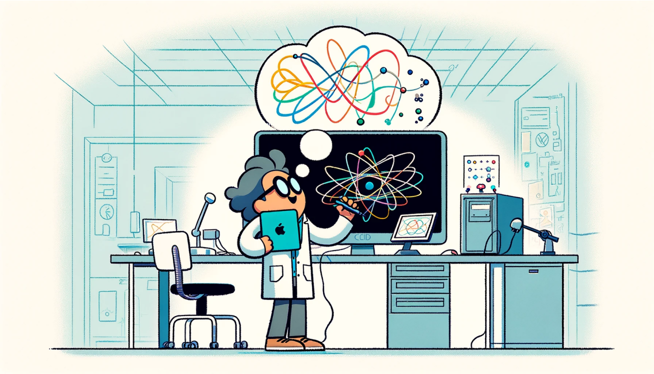
- qw5q from QuantWare and controlled using Qblox instruments;
- iqm5q from IQM and controlled using Zurich Instruments.

Train.	Epochs	Pred.	Config.	MSE
qw5q	50	qw5q	noisy	0.0055
qw5q	50	qw5q	RTQEM	0.0042
qw5q	100	qw5q	RTQEM	0.0013
iqm5q	100	qw5q	RTQEM	0.0037
qw5q	100	sim	RTQEM	0.0016

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

Outlook

- Can improve RTQEM robustness? Is ICS enough to face any scenario?
- how about combining more techniques?
- how about adding some memory (inertia) to the noise model? Can this be robust against sudden (and temporary) system's fluctuations?
- how about moving the mitigation routine on chip to boost the process?



DALLE, please, draw a PhD student trying hard to fit Parton Distribution Functions on a quantum computer at CERN