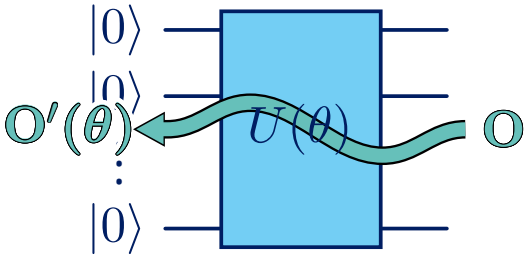
Training of Large Shallow Parametrized Quantum Circuits through Pauli Propagation

LHC Quantum Computing Workshop

Saverio Monaco

September 24, 2026



QUANTUM TECHNOLOGY INITIATIVE

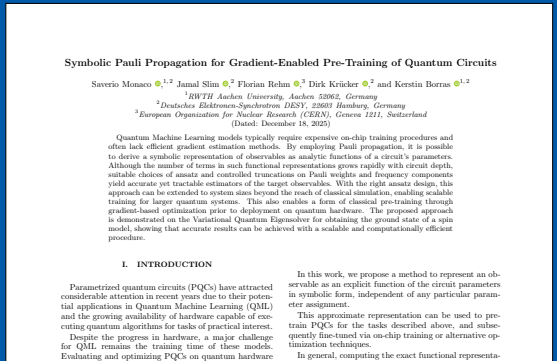
Outline

1. Introduction
2. Propagation of Observables
3. Applications

arXiv: https://arxiv.org/abs/2512.16674

https://github.com/desyqml/Pauli-Propagator

Try it online via Binder: launch binder



Symbolic Pauli Propagation for Gradient-Enabled Pre-Training of Quantum Circuits

Saverio Monaco [1,2] Jamal Slim [2] Florian Rehn [3] Dirk Krücker [2] and Kerstin Borras [1,2]
[1] RWTH Aachen University, Aachen 52062, Germany
[2] Deutsches Elektronen-Synchrotron DESY, 22603 Hamburg, Germany
[3] European Organization for Nuclear Research (CERN), Geneva 1211, Switzerland
(Dated: December 18, 2025)

Quantum Machine Learning models typically require expensive on-chip training procedures and often lack efficient gradient estimation methods. By employing Pauli propagation, it is possible to derive a symbolic representation of observables as analytic functions of a circuit's parameters. Although the number of terms in such functional representations grows rapidly with circuit depth, suitable choices of ansatz and controlled truncations on Pauli weights and frequency components yield accurate yet tractable estimators of the target observables. With the right ansatz design, this approach can be extended to system sizes beyond the reach of classical simulation, enabling scalable training for larger quantum systems. This also enables a form of classical pre-training through gradient-based optimization prior to deployment on quantum hardware. The proposed approach is demonstrated on the Variational Quantum Eigensolver for obtaining the ground state of a spin model, showing that accurate results can be achieved with a scalable and computationally efficient procedure.

I. INTRODUCTION

Parametrized quantum circuits (PQCs) have attracted considerable attention in recent years due to their potential applications in Quantum Machine Learning (QML) and the growing availability of hardware capable of executing quantum algorithms for tasks of practical interest.

Despite the progress in hardware, a major challenge for QML remains the training time of these models. Evaluating and optimizing PQCs on quantum hardware

In this work, we propose a method to represent an observable as an explicit function of the circuit parameters in symbolic form, independent of any particular parameter assignment.

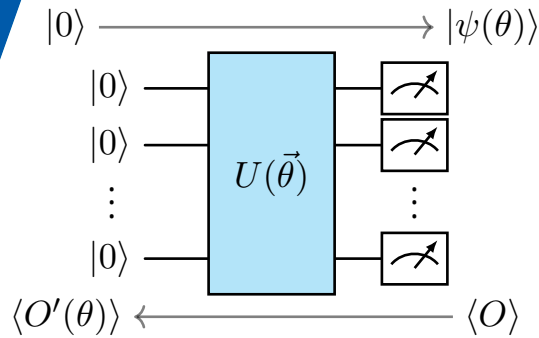
This approximate representation can be used to pre-train PQCs for the tasks described above, and subsequently fine-tuned via on-chip training or alternative optimization techniques.

In general, computing the exact functional representa-

QUANTUM TECHNOLOGY INITIATIVE

Introduction

Setting and Objectives



Introduction

Problem: training large parametrized quantum circuits is tricky

- **Access to hardware:** quantum devices are a scarce, costly and queued resource
- **Gradient-free optimizers** (SPSA [1], COBYLA [2]) reduce the number of circuit evaluations, but offer weak convergence guarantees and require hardware access
- **Parameter-shift rule** [3] gives exact gradients, but scales poorly: $2 \times N_{\text{param}} \times N_{\text{shots}}$ circuit evaluations

Example: time to evaluate the gradient of the MMD loss for a **single training step** on MNIST ($N_{\text{param}} = 10\,000$, precision $\epsilon = 10^{-3}$, training set N_{images} = 60,000) [4]:

$$2 \times N_{\text{param}} \times \epsilon^{-2} \times N_{\text{images}} = 1.2 \times 10^{15} \text{ shots}$$

Introduction

Problem: training large parametrized quantum circuits is tricky

- **Access to hardware:** quantum devices are a scarce, costly and queued resource
- **Gradient-free optimizers** (SPSA [1], COBYLA [2]) reduce the number of circuit evaluations, but offer weak convergence guarantees and require hardware access
- **Parameter-shift rule** [3] gives exact gradients, but scales poorly: $2 \times N_{\text{param}} \times N_{\text{shots}}$ circuit evaluations

Example: time to evaluate the gradient of the MMD loss for a **single training step** on MNIST ($N_{\text{param}} = 10\,000$, precision $\epsilon = 10^{-3}$, training set N_{images} = 60,000) [4]:

$$2 \times N_{\text{param}} \times \epsilon^{-2} \times N_{\text{images}} = 1.2 \times 10^{15} \text{ shots}$$

$$\Rightarrow \approx \textbf{38 years} \text{ (at } 10^6 \text{ shots/s)}$$

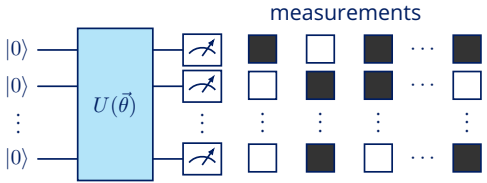
Introduction

Claim: Expectation values of a quantum circuit can be estimated classically*

*up to some accuracy levels

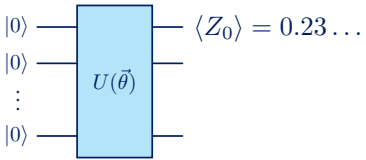
**accuracy guarantees limited to a class of quantum circuits (*locally-scrambling*)

Accessing the wavefunction



Classically hard

Computing expectation value

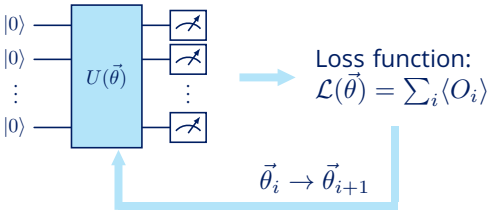


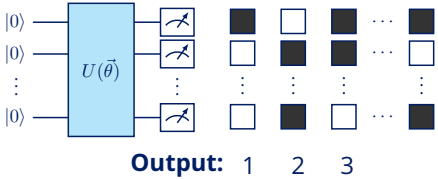
Classically tractable*

Introduction

Potential applications: algorithms whose output is a single wavefunction collapse (measurement), but the loss function can be formalized through expectation values:

Training on classical resources (Pauli Propagation) → Deployment on quantum resources



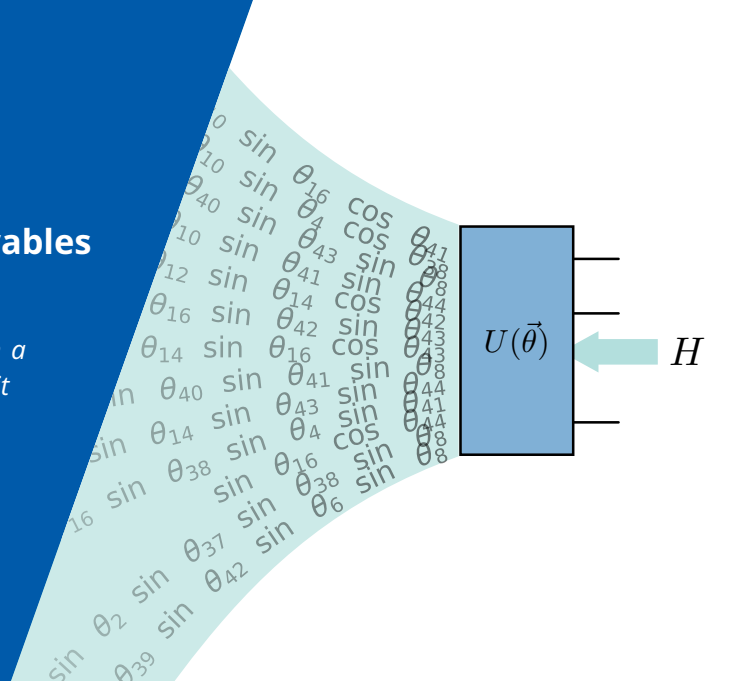


Applications: Variational Quantum Eigensolver [5], Compression (Autoencoder) [6], Generation [7, 8]

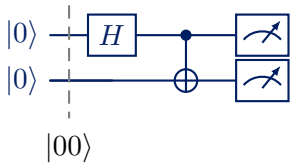
QUANTUM TECHNOLOGY INITIATIVE

Propagation of Observables

Evolving Observables through a Parametrized Quantum Circuit



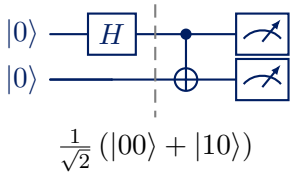
Propagation



Schrödinger Picture:
Propagate the Wavefunction through the circuit according to the rule:

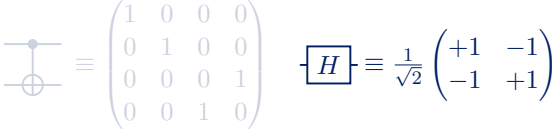
$$|\psi\rangle' = U|\psi\rangle$$

Propagation

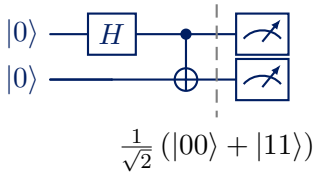


Schrödinger Picture:
Propagate the Wavefunction through the circuit according to the rule:

$$|\psi\rangle' = U|\psi\rangle$$



Propagation



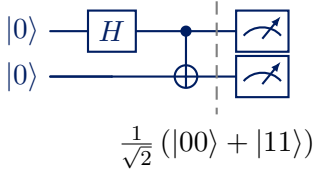
Schrödinger Picture:
Propagate the Wavefunction through the circuit according to the rule:

$$|\psi\rangle' = U|\psi\rangle$$

$$\equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\equiv \frac{1}{\sqrt{2}} \begin{pmatrix} +1 & -1 \\ -1 & +1 \end{pmatrix}$$

Propagation



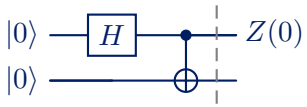
Schrödinger Picture:
Propagate the Wavefunction through the circuit according to the rule:

$$|\psi\rangle' = U|\psi\rangle$$

$$\equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} +1 & -1 \\ -1 & +1 \end{pmatrix}$$

Example: $\langle Z(0) \rangle = \langle \psi | Z\, I\, | \psi \rangle = 0$

Propagation



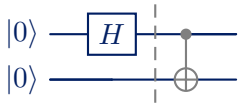
Heisenberg Picture:
Evolve the operator *"backwards"* through the circuit according to the rule:

$$V' = U^\dagger V U$$

(Similarity transformation)

$Z(0)I(1)$

Propagation



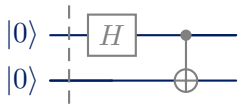
Heisenberg Picture:
Evolve the operator *"backwards"* through the circuit according to the rule:

$$V' = U^\dagger V U$$

(Similarity transformation)

$$Z(0)I(1) = CNOT^\dagger Z(0)I(1)CNOT$$

Propagation



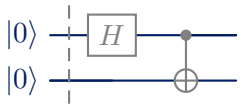
Heisenberg Picture:
Evolve the operator *"backwards"* through the circuit according to the rule:

$$V' = U^\dagger V U$$

(Similarity transformation)

$$X(0)I(1) = H^\dagger Z(0)I(1)H$$

Propagation



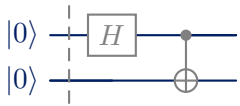
Heisenberg Picture:
Evolve the operator *"backwards"* through the circuit according to the rule:

$$V' = U^\dagger V U$$

(Similarity transformation)

$$\langle 00|X(0)I(1)|00\rangle = 0$$

Propagation



Heisenberg Picture:
Evolve the operator *"backwards"* through the circuit according to the rule:

$$V' = U^\dagger V U$$

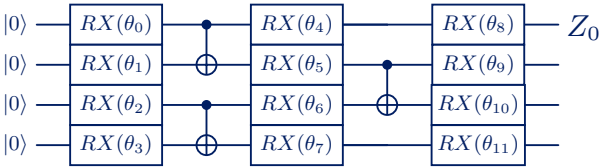
(Similarity transformation)

$$\langle 00|X(0)I(1)|00\rangle = 0 \rightarrow \langle 00|U^\dagger Z(0)U|00\rangle = 0$$

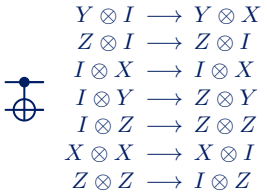
Propagation

Paper example

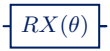
Step-by-step propagation:



Propagation tables:



$$
\begin{aligned}
Y \otimes I &\longrightarrow Y \otimes X \\
Z \otimes I &\longrightarrow Z \otimes I \\
I \otimes X &\longrightarrow I \otimes X \\
I \otimes Y &\longrightarrow Z \otimes Y \\
I \otimes Z &\longrightarrow Z \otimes Z \\
X \otimes X &\longrightarrow X \otimes I \\
Z \otimes Z &\longrightarrow I \otimes Z
\end{aligned}
$$

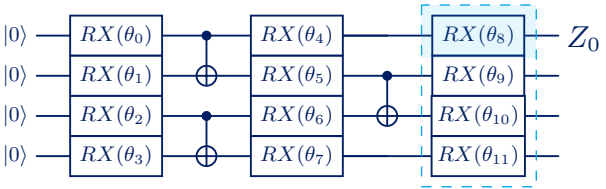


$$
\begin{aligned}
I &\longrightarrow I \\
X &\longrightarrow X \\
Y &\longrightarrow +\cos\theta Y \\
&\quad\ -\sin\theta Z \\
Z &\longrightarrow +\cos\theta Z \\
&\quad\ +\sin\theta Y
\end{aligned}
$$

Propagation

Paper example

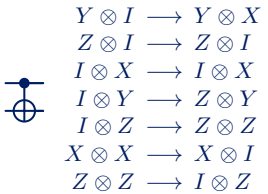
Step-by-step propagation:



- **Layer 5:** At this stage, the observable Z_0 is only affected by the gate $RX(\theta_8)$. This evolves as:

$$Z_0 \rightarrow \cos\theta_8 Z_0 + \sin\theta_8 Y_0$$

Propagation tables:

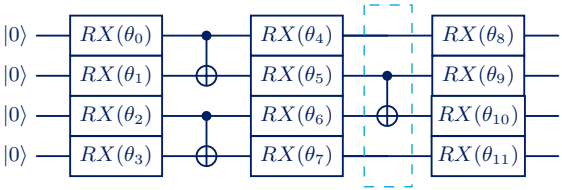


$$\begin{aligned} Y \otimes I &\longrightarrow Y \otimes X \\ Z \otimes I &\longrightarrow Z \otimes I \\ I \otimes X &\longrightarrow I \otimes X \\ I \otimes Y &\longrightarrow Z \otimes Y \\ I \otimes Z &\longrightarrow Z \otimes Z \\ X \otimes X &\longrightarrow X \otimes I \\ Z \otimes Z &\longrightarrow I \otimes Z \end{aligned}$$

$$\begin{aligned} I &\longrightarrow I \\ X &\longrightarrow X \\ Y &\longrightarrow +\cos\theta Y \\ &\quad\ -\sin\theta Z \\ Z &\longrightarrow +\cos\theta Z \\ &\quad\ +\sin\theta Y \end{aligned}$$

Propagation

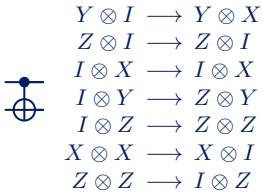
Step-by-step propagation:



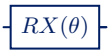
- **Layer 4:** Both terms of the observable commute with CNOT$(1, 2)$, hence no transformation occurs:

$$\cos\theta_8 Z_0 + \sin\theta_8 Y_0 \rightarrow \cos\theta_8 Z_0 + \sin\theta_8 Y_0$$

Propagation tables:



$$\begin{aligned} Y \otimes I &\longrightarrow Y \otimes X \\ Z \otimes I &\longrightarrow Z \otimes I \\ I \otimes X &\longrightarrow I \otimes X \\ I \otimes Y &\longrightarrow Z \otimes Y \\ I \otimes Z &\longrightarrow Z \otimes Z \\ X \otimes X &\longrightarrow X \otimes I \\ Z \otimes Z &\longrightarrow I \otimes Z \end{aligned}$$

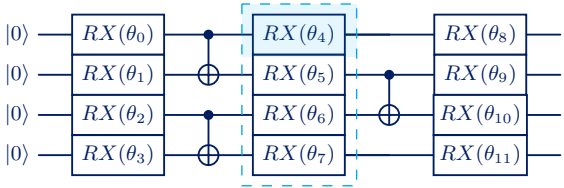


$$\begin{aligned} I &\longrightarrow I \\ X &\longrightarrow X \\ Y &\longrightarrow +\cos\theta Y \\ &\quad\ -\sin\theta Z \\ Z &\longrightarrow +\cos\theta Z \\ &\quad\ +\sin\theta Y \end{aligned}$$

Propagation

Paper example

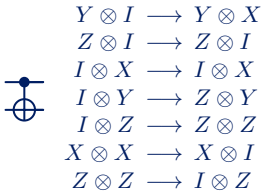
Step-by-step propagation:



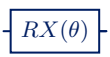
- **Layer 3:** Here only the gate $RX(\theta_4)$ affects the propagated observable:

$$\begin{aligned}\cos\theta_8 Z_0 + \sin\theta_8 Y_0 \rightarrow &+ \cos\theta_8 \cos\theta_4 Z_0 \\ &+ \cos\theta_8 \sin\theta_4 Y_0 \\ &+ \sin\theta_8 \cos\theta_4 Y_0 \\ &- \sin\theta_8 \sin\theta_4 Z_0\end{aligned}$$

Propagation tables:



$$\begin{aligned}Y \otimes I &\longrightarrow Y \otimes X \\ Z \otimes I &\longrightarrow Z \otimes I \\ I \otimes X &\longrightarrow I \otimes X \\ I \otimes Y &\longrightarrow Z \otimes Y \\ I \otimes Z &\longrightarrow Z \otimes Z \\ X \otimes X &\longrightarrow X \otimes I \\ Z \otimes Z &\longrightarrow I \otimes Z\end{aligned}$$

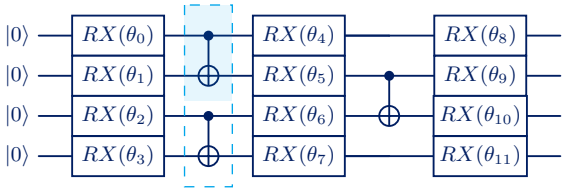


$$\begin{aligned}I &\longrightarrow I \\ X &\longrightarrow X \\ Y &\longrightarrow +\cos\theta Y \\ &\quad\ -\sin\theta Z \\ Z &\longrightarrow +\cos\theta Z \\ &\quad\ +\sin\theta Y\end{aligned}$$

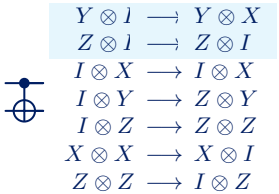
Propagation

Paper example

Step-by-step propagation:



Propagation tables:



$$
\begin{aligned}
Y \otimes 1 &\longrightarrow Y \otimes X \\
Z \otimes 1 &\longrightarrow Z \otimes I \\
I \otimes X &\longrightarrow I \otimes X \\
I \otimes Y &\longrightarrow Z \otimes Y \\
I \otimes Z &\longrightarrow Z \otimes Z \\
X \otimes X &\longrightarrow X \otimes I \\
Z \otimes Z &\longrightarrow I \otimes Z
\end{aligned}
$$

$$
\begin{aligned}
I &\longrightarrow I \\
X &\longrightarrow X \\
Y &\longrightarrow +\cos\theta Y \\
&\quad\ -\sin\theta Z \\
Z &\longrightarrow +\cos\theta Z \\
&\quad\ +\sin\theta Y
\end{aligned}
$$

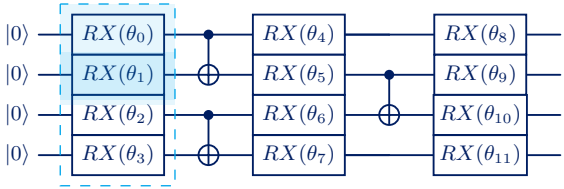
- **Layer 2:** The CNOT$(0,1)$ gate acts on all observables involving the first two qubits:

$$
\begin{aligned}
&+\cos\theta_8\cos\theta_4 Z_0 + \cos\theta_8\sin\theta_4 Y_0 \\
&+\sin\theta_8\cos\theta_4 Y_0 - \sin\theta_8\sin\theta_4 Z_0 \\
&\quad\downarrow \\
&+\cos\theta_8\cos\theta_4 Z_0 + \cos\theta_8\sin\theta_4 Y_0 X_1 \\
&+\sin\theta_8\cos\theta_4 Y_0 X_1 - \sin\theta_8\sin\theta_4 Z_0
\end{aligned}
$$

Propagation

Paper example

Step-by-step propagation:

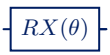


- **Layer 1:** In this final layer, both $RX(\theta_0)$ and $RX(\theta_1)$ act on the observable.
  - **Action of $RX(\theta_1)$:** Every term's qubit-1 factor is either I_1 or X_1; both are invariant under RX, so the word is unchanged:

$$
\begin{aligned}
&+\cos\theta_8\cos\theta_4 Z_0 + \cos\theta_8\sin\theta_4 Y_0X_1\\
&+\sin\theta_8\cos\theta_4 Y_0X_1 - \sin\theta_8\sin\theta_4 Z_0\\
&\quad\downarrow\\
&+\cos\theta_8\cos\theta_4 Z_0 + \cos\theta_8\sin\theta_4 Y_0X_1\\
&+\sin\theta_8\cos\theta_4 Y_0X_1 - \sin\theta_8\sin\theta_4 Z_0
\end{aligned}
$$

Propagation tables:

$$
\begin{aligned}
Y\otimes I &\longrightarrow Y\otimes X\\
Z\otimes I &\longrightarrow Z\otimes I\\
I\otimes X &\longrightarrow I\otimes X\\
I\otimes Y &\longrightarrow Z\otimes Y\\
I\otimes Z &\longrightarrow Z\otimes Z\\
X\otimes X &\longrightarrow X\otimes I\\
Z\otimes Z &\longrightarrow I\otimes Z
\end{aligned}
$$

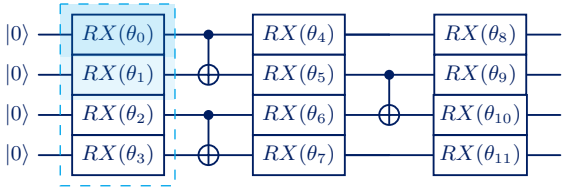


$$
\begin{aligned}
I &\longrightarrow I\\
X &\longrightarrow X\\
Y &\longrightarrow +\cos\theta Y\\
&\quad\ -\sin\theta Z\\
Z &\longrightarrow +\cos\theta Z\\
&\quad\ +\sin\theta Y
\end{aligned}
$$

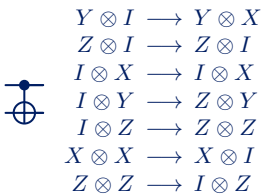
Propagation

Paper example

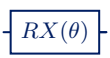
Step-by-step propagation:



Propagation tables:



$$
\begin{aligned}
Y \otimes I &\longrightarrow Y \otimes X \\
Z \otimes I &\longrightarrow Z \otimes I \\
I \otimes X &\longrightarrow I \otimes X \\
I \otimes Y &\longrightarrow Z \otimes Y \\
I \otimes Z &\longrightarrow Z \otimes Z \\
X \otimes X &\longrightarrow X \otimes I \\
Z \otimes Z &\longrightarrow I \otimes Z
\end{aligned}
$$



$$
\begin{aligned}
I &\longrightarrow I \\
X &\longrightarrow X \\
Y &\longrightarrow +\cos\theta Y \\
&\quad\ -\sin\theta Z \\
Z &\longrightarrow +\cos\theta Z \\
&\quad\ +\sin\theta Y
\end{aligned}
$$

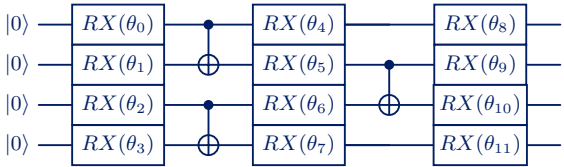
- **Layer 1:** In this final layer, both $RX(\theta_0)$ and $RX(\theta_1)$ act on the observable.
  - **Action of $RX(\theta_0)$:**

$$
\begin{aligned}
&+\cos\theta_8\cos\theta_4 Z_0 + \cos\theta_8\sin\theta_4 Y_0 X_1 \\
&+\sin\theta_8\cos\theta_4 Y_0 X_1 - \sin\theta_8\sin\theta_4 Z_0 \\
&\quad\downarrow \\
&+\cos\theta_8\cos\theta_4\cos\theta_0 Z_0 \\
&+\cos\theta_8\cos\theta_4\sin\theta_0 Y_0 \\
&+\cos\theta_8\sin\theta_4\cos\theta_0 Y_0 X_1 \\
&-\cos\theta_8\sin\theta_4\sin\theta_0 Z_0 X_1 \\
&+\sin\theta_8\cos\theta_4\cos\theta_0 Y_0 X_1 \\
&-\sin\theta_8\cos\theta_4\sin\theta_0 Z_0 X_1 \\
&-\sin\theta_8\sin\theta_4\cos\theta_0 Z_0 \\
&-\sin\theta_8\sin\theta_4\sin\theta_0 Y_0
\end{aligned}
$$

Propagation

Paper example

Step-by-step propagation:



Expectation value rules:

$$\langle 0| I |0\rangle = 1 \qquad \langle 0| Y |0\rangle = 0$$

$$\langle 0| X |0\rangle = 0 \qquad \langle 0| Z |0\rangle = 1$$

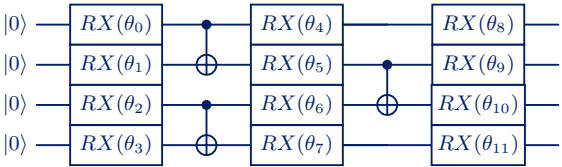
- **Expectation value:** Projecting onto $|0\rangle^{\otimes 4}$, only words made of I and Z survive:

$$+\cos\theta_8\cos\theta_4\cos\theta_0 Z_0$$
$$+\cancel{\cos\theta_8\cos\theta_4\sin\theta_0 Y_0}$$
$$+\cancel{\cos\theta_8\sin\theta_4\cos\theta_0 Y_0 X_1}$$
$$-\cancel{\cos\theta_8\sin\theta_4\sin\theta_0 Z_0 X_1}$$
$$+\cancel{\sin\theta_8\cos\theta_4\cos\theta_0 Y_0 X_1}$$
$$-\cancel{\sin\theta_8\cos\theta_4\sin\theta_0 Z_0 X_1}$$
$$-\sin\theta_8\sin\theta_4\cos\theta_0 Z_0$$
$$-\cancel{\sin\theta_8\sin\theta_4\sin\theta_0 Y_0}$$

Propagation

Paper example

Step-by-step propagation:



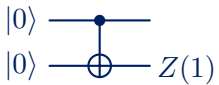
Functional form:

$$\langle 0|Z_0|0\rangle(\vec{\theta}) = +\cos\theta_8\cos\theta_4\cos\theta_0 - \sin\theta_8\sin\theta_4\cos\theta_0$$

Propagation example

Three notable effects happen when propagating through a circuit:

 # 1: Weight increase:
Any non-commuting **multi-qubit gate** increases the weight:



Weight: $\underbrace{Z(1)}_{1}$

$$
\begin{aligned}
I \otimes I &\longrightarrow I \otimes I \\
I \otimes X &\longrightarrow I \otimes X \\
I \otimes Y &\longrightarrow Z \otimes Y \\
I \otimes Z &\longrightarrow Z \otimes Z
\end{aligned}
$$

Propagation example

Three notable effects happen when propagating through a circuit:

 # 1: Weight increase:
Any non-commuting **multi-qubit gate** increases the weight:

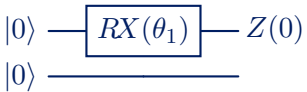
Weight: $\underbrace{Z(0)Z(1)}_{2}$

$$
\begin{aligned}
I \otimes I &\longrightarrow I \otimes I \\
I \otimes X &\longrightarrow I \otimes X \\
I \otimes Y &\longrightarrow Z \otimes Y \\
I \otimes Z &\longrightarrow Z \otimes Z
\end{aligned}
$$

Propagation example

Two notable effects happen when propagating through a circuit:

 # 2: Branching & # 3: Frequency increase:
Any non-commuting non-clifford gate splits the term into two and any non-commuting **parametrized gate** increases the frequency:



Frequencies: $\underbrace{Z(0)}_{0}$

$$
\begin{aligned}
I &\longrightarrow I \\
X &\longrightarrow X \\
Y &\longrightarrow +\cos(\theta)Y - \sin(\theta)Z \\
Z &\longrightarrow +\cos(\theta)Z + \sin(\theta)Y
\end{aligned}
$$

Propagation example

Two notable effects happen when propagating through a circuit:

 # 2: Branching & # 3: Frequency increase:
Any non-commuting non-clifford gate splits the term into two and any non-commuting **parametrized gate** increases the frequency:

$|0\rangle$ — $\cos(\theta_1)Z(0) + \sin(\theta_1)Y(0)$

$|0\rangle$ —

Frequencies: $\underbrace{\cos(\theta_1)Z(0)}_{1} + \underbrace{\sin(\theta_1)Y(0)}_{1}$

$RX(\theta)$

$$
\begin{aligned}
I &\longrightarrow I \\
X &\longrightarrow X \\
Y &\longrightarrow +\cos(\theta)Y - \sin(\theta)Z \\
Z &\longrightarrow +\cos(\theta)Z + \sin(\theta)Y
\end{aligned}
$$

Exact propagation

Scaling at increasing depth: branching causes the doubling of every word for each non-commuting gate $\rightarrow$ easily intractable!

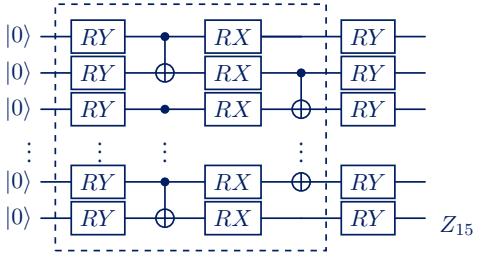


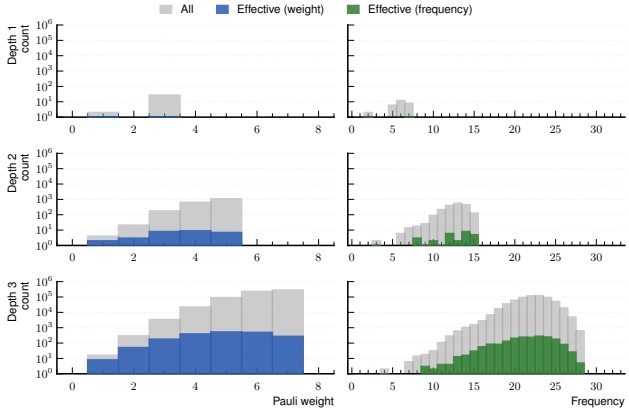
Figure: Local entangler Ansatz at depth 1 (iterations of the unitaries in the dashed block)

Small circuit to perform exact propagation of observable Z_{15}.
Number of qubits: 16.

Exact propagation

Small circuit example

Distribution of final weights and frequencies at different depths (1-3):



Example: $\underbrace{\sin\theta_8 \sin\theta_1 \cos\theta_0}_{\text{Frequency } (3)}$ $\underbrace{X_0 Z_1}_{\text{Weight } (2)}$

- Deeper circuits lead to more branching, doubling the number of Pauli words.
- Most Pauli words do not contribute to the expectation value of the observable and are discarded during the trimming process (gray).
- Branching causes a large number of Pauli words to be tracked, making cutoffs necessary.

Cutoffs

Example: $\underbrace{\sin\theta_8 \sin\theta_1 \cos\theta_0}_{\text{Frequency } (3)} \underbrace{X_0 Z_1}_{\text{Weight } (2)}$

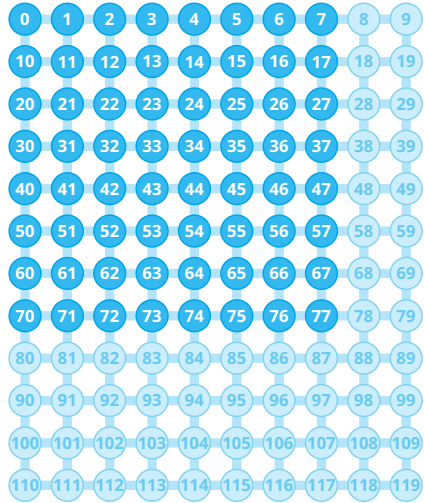
Cutoff strategy: fix a max weight w_{cut} and a max frequency ν_{cut}; while propagating the observable, track the weight and frequency of every Pauli word and discard any that exceed them.

- **Weight truncation:** for **locally-scrambling** ansätze, the contribution of a path to the expectation value decays **exponentially** with its terminal weight (theoretical guarantee [9]).
- **Frequency truncation:** each branching multiplies the coefficient by a $\sin\theta_k$ or $\cos\theta_k$ factor, so a term's contribution decays exponentially with its frequency (assuming uniformly-distributed parameters [10]).

QUANTUM TECHNOLOGY INITIATIVE

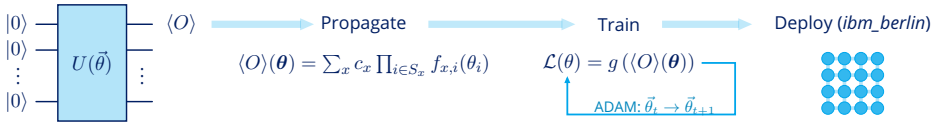
Applications

Example application of Symbolic Propagation: Variational Quantum Eigensolver, and MNIST Digits Generation



Applications

Outline in QML:



Step 1: Define the ansatz and express the loss function as expectation values of Pauli observables.

Step 2: Apply Pauli Propagation to obtain an explicit expression for the expectation value in terms of the circuit parameters.

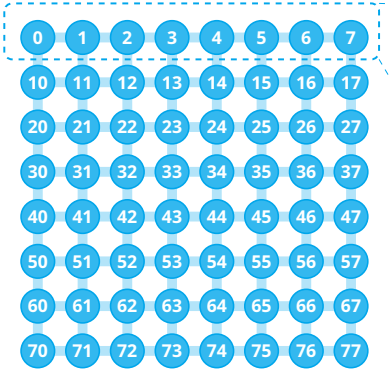
Step 3: Apply ADAM to optimize the circuit parameters classically.

Step 4: Deploy the optimized circuit on a quantum device.

- Low cutoff values $\rightarrow$ fast propagation times, approximate training, warm start;
- High cutoff values $\rightarrow$ slow propagation times, accurate training.

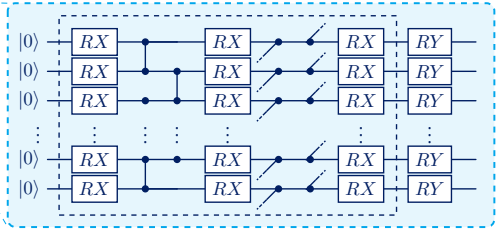
Applications

Employed quantum hardware



ibm_berlin Quantum Computer

Employed ansatz



Number of qubits: 64

Number of parameters:

256 (depth 1) | 384 (depth 2)

Two-qubit circuit depth (transpiled)

4 (depth 1) | 8 (depth 2)

Application # 1

Spin Model: 2D Square lattice (8x8) with nearest neighbor interactions and magnetic field

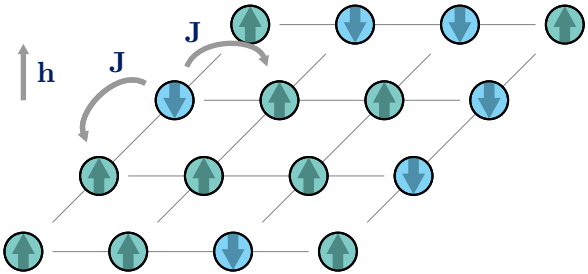
Goal: Get the ground state of a Hamiltonian H.

Loss: $E(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle$

Number of qubits: 64

Hamiltonian:

$$H(J,h) = \frac{1}{N}\Big(- J \sum_{\langle i,j \rangle} Z(i)Z(j) - h \sum_{i} X(i) \Big)$$



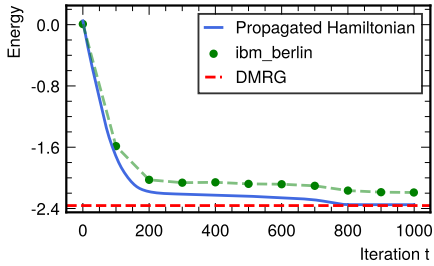
Application # 1

Training run # 1:

- **Depth:** 1
- **Cutoffs:**
  - $w_{\text{cut}} = \text{None}$
  - $\nu_{\text{cut}} = \text{None}$ (Exact propagation)

DMRG ground state: $E = -2.3618$

	Final energy	Gap to DMRG
Propagated	-2.3504	0.0114
ibm_berlin	-2.1910	0.1708



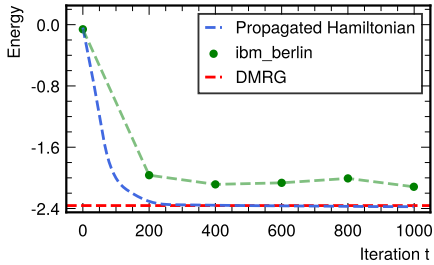
Application # 1

Training run # 2:

- **Depth:** 2
- **Cutoffs:**
  - $w_{\text{cut}} = 12$
  - $\nu_{\text{cut}} = \text{None}$

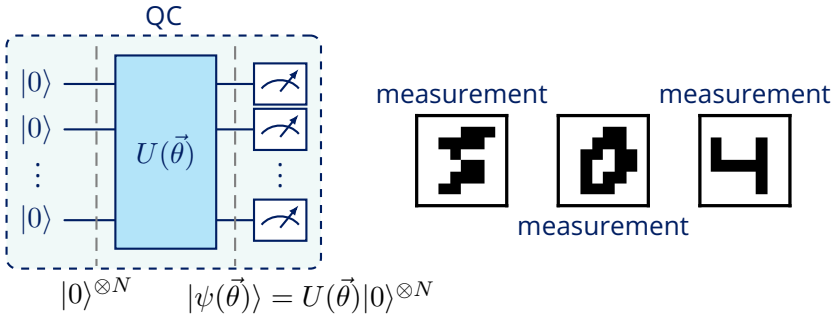
DMRG ground state: $E = -2.3618$

	Final energy	Gap to DMRG
Propagated	-2.3765	0.0147
ibm_berlin	-2.1165	0.2453



Application # 2

Goal: Training of a sampling-type generative model: the squared norm of the evolved wavefunction, $|\psi(\boldsymbol{\theta})(x)|^2$, should be close to the target distribution of the training data, $p_{\text{data}}(x)$.



1 qubit $\leftrightarrow$ 1 pixel correspondence, binary images: white $\equiv$ spin-down, black $\equiv$ spin-up.

Application # 2

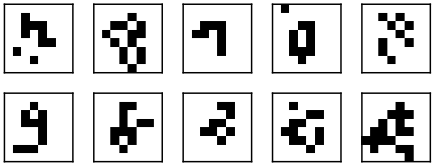
Training run:

- **Depth:** 1
- **Cutoffs:** $w_{\text{cut}} = 12$, $\nu_{\text{cut}} = \text{None}$

MMD on test set (mean $\pm$ std, 100 draws):

Model	MMD
Trained ansatz (256 params)	0.00285 ± 0.00021
IQP-1x (256 params)	0.02572 ± 0.00139
IQP-10x (2560 params)	0.00431 ± 0.00038

Result:



Shallow Ansatz + Limited hardware + Complex task:
evolving the state into a superposition of all possible images!

Final words

Pauli Propagation: classically estimates **expectation values of observables** of a parametrized quantum circuit.

What it enables: **pre-training (warm-starting)**, or even **full training**, of QML models whose loss is an expectation value, even when the quantity of interest itself (full state, measurements, …) is **quantum-hard** to obtain classically.

Limitations: deep ansätze lead to **long propagation times**; cutoffs trade exactness for tractability.

Outlook: find **families of ansätze and observables** that are both **useful** and **efficient to propagate**, e.g. by exploiting **geometric structure** [11].

QUANTUM TECHNOLOGY INITIATIVE

Main resources:

- **Paper**: "Classically estimating observables of noiseless quantum circuits"
  `https://arxiv.org/abs/2409.01706`
- **Paper**: "The Heisenberg Representation of Quantum Computers"
  `https://arxiv.org/abs/quant-ph/9807006`
- **Pennylane Demo**: "Pauli Propagation: Classically estimating expectation values from parametrized quantum circuits"
  `https://pennylane.ai/demos/tutorial_classical_expval_estimation`

DESY. QUANTUM

RWTH AACHEN UNIVERSITY

ENGAGE has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie Grant Agreement No. 101034267.

QUANTUM TECHNOLOGY INITIATIVE

Thank you

DESY. QUANTUM

RWTH AACHEN UNIVERSITY

ENGAGE has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie Grant Agreement No. 101034267.

QUANTUM TECHNOLOGY INITIATIVE

References I

[1] James C Spall.
An overview of the simultaneous perturbation method for efficient optimization.
Johns Hopkins apl technical digest, 19(4):482–492, 1998.

[2] Michael JD Powell.
A direct search optimization method that models the objective and constraint functions by linear interpolation.
In *Advances in optimization and numerical analysis*, pages 51–67. Springer, 1994.

[3] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii.
Quantum circuit learning, 2019.

[4] Erik Recio-Armengol, Shahnawaz Ahmed, and Joseph Bowles.
Train on classical, deploy on quantum: scaling generative quantum machine learning to a thousand qubits, 2025.

QUANTUM TECHNOLOGY INITIATIVE

References II

[5] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien.
A variational eigenvalue solver on a quantum processor, 2013.

[6] Korbinian Kottmann, Friederike Metz, Joana Fraxanet, and Niccolo Baldelli.
Variational quantum anomaly detection: Unsupervised mapping of phase diagrams on a physical quantum computer, 2022.

[7] Erik Recio-Armengol and Joseph Bowles.
Iqpopt: Fast optimization of instantaneous quantum polynomial circuits in jax, 2025.

[8] Manuel S. Rudolph, Sacha Lerch, Supanut Thanasilp, Oriel Kiss, Sofia Vallecorsa, Michele Grossi, and Zoë Holmes.
Trainability barriers and opportunities in quantum generative modeling, 2023.

QUANTUM TECHNOLOGY INITIATIVE

References III

[9] Armando Angrisani, Alexander Schmidhuber, Manuel S. Rudolph, M. Cerezo, Zoë Holmes, and Hsin-Yuan Huang.
Classically estimating observables of noiseless quantum circuits, 2025.

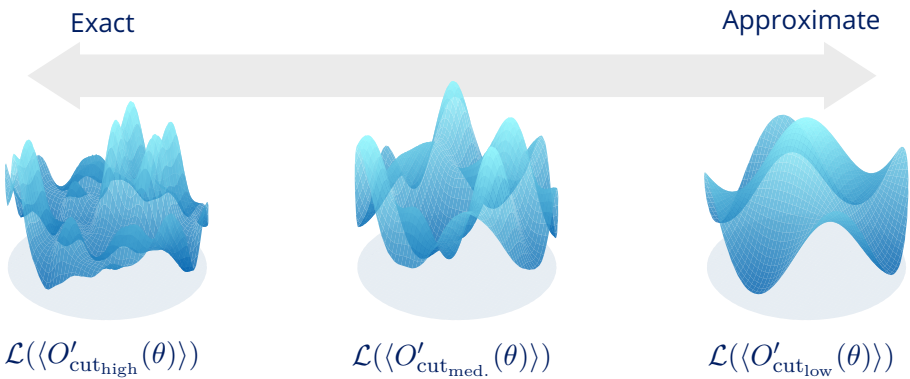
[10] Saverio Monaco, Jamal Slim, Florian Rehm, Dirk Krücker, and Kerstin Borras.
Symbolic pauli propagation for gradient-enabled pre-training of quantum circuits, 2025.

[11] Yanting Teng, Su Yeon Chang, Manuel S. Rudolph, and Zoë Holmes.
Leveraging symmetry merging in pauli propagation, 2025.

[12] Armando Angrisani, Antonio A. Mele, Manuel S. Rudolph, M. Cerezo, and Zoë Holmes.
Simulating quantum circuits with arbitrary local noise using pauli propagation, 2025.

Cutoffs

Cutoffs produce an approximate expectation value of the observable, and training on this approximation may still yield valuable results, though not fully optimal

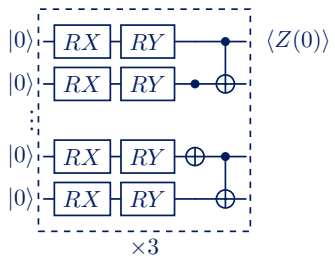


Locally-scrambling circuits

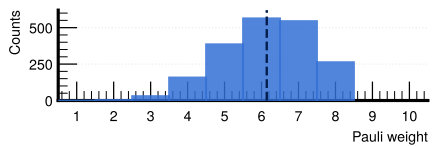
Definition: a circuit is *locally-scrambling* [9] if

- **Rotations:** the single-qubit rotation gates are not all aligned along one axis, but random enough on each qubit that no direction (X, Y, Z) is systematically preferred.
- **Connectivity:** the entangling gates are local, acting only between nearby qubits.

Example: Locally-scrambling circuit



Distribution of weights

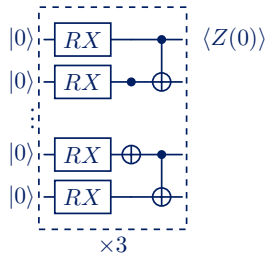


Locally-scrambling circuits

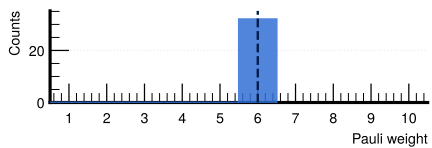
Definition: a circuit is *locally-scrambling* [9] if

- **Rotations:** the single-qubit rotation gates are not all aligned along one axis, but random enough on each qubit that no direction (X, Y, Z) is systematically preferred.
- **Connectivity:** the entangling gates are local, acting only between nearby qubits.

Example: Non-scrambling circuit



Distribution of weights

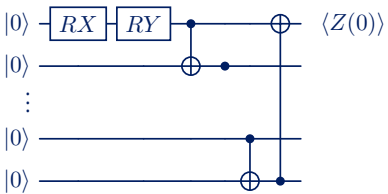


Locally-scrambling circuits

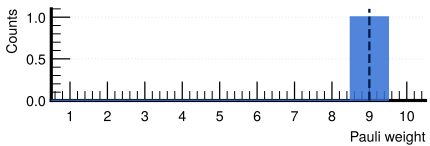
Definition: a circuit is *locally-scrambling* [9] if

- **Rotations:** the single-qubit rotation gates are not all aligned along one axis, but random enough on each qubit that no direction (X, Y, Z) is systematically preferred.
- **Connectivity:** the entangling gates are local, acting only between nearby qubits.

Example: Non-local circuit



Distribution of weights



Propagation

Non-Clifford Gates:

$$RX(\theta) \equiv \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$RY(\theta) \equiv \begin{pmatrix} +\cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ -\sin(\frac{\theta}{2}) & +\cos(\frac{\theta}{2}) \end{pmatrix}$$

$$RZ(\theta) \equiv \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{+i\theta/2} \end{pmatrix}$$

Propagation causes **branchings** and **frequency** increase.

Effects:

- **Branching**:
  - $RY(\theta)^\dagger \mathbf{X}(\mathbf{1}) RY(\theta)$
    $= \cos(\theta) X(1) + \sin(\theta) Z(1).$

  Branching causes non-commuting words to double.

- **Frequency increase:**
  - $RY(\theta_2)^\dagger (\mathbf{cos}(\boldsymbol{\theta_1})\mathbf{X}(\mathbf{1})) RY(\theta_2)$
    $= \cos(\theta_2)\cos(\theta_1) X(1)$
    $+ \sin(\theta_2)\cos(\theta_1) Z(1).$

Propagation

Clifford Gates:


$\equiv \frac{1}{\sqrt{2}} \begin{pmatrix} +1 & -1 \\ -1 & +1 \end{pmatrix}$


$\equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Clifford Gates **do not** cause *branchings* or *frequency increase*.

Effects:

- **Weight increase**:
  - $CX^\dagger I(1)Z(2)CX = Z(1)Z(2).$

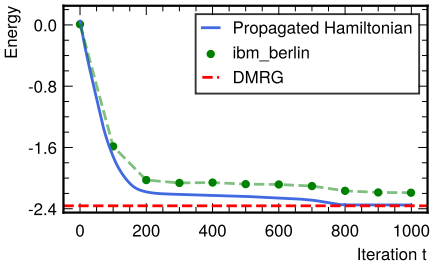
  Multi-qubit gates can increase the weight of the Pauli word.

- **Simplest propagation:**
  - $H^\dagger X(1)H = Z(1).$

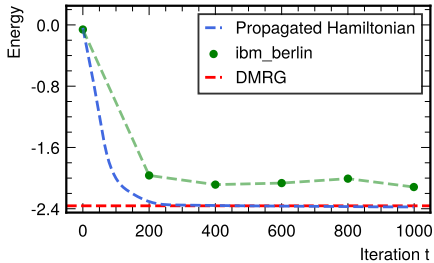
  Does not increase the weight of the Pauli word, and does not cause branching or frequency increase.

Application # 1

Depth 1:



Depth 2:



Gap between expectation value of the propagated Hamiltonian and actual value on *ibm_berlin* is likely due to residual hardware noise.

Query on *ibm_berlin*: evaluating $\langle H \rangle$ with 4096 shots at resilience level 2 (readout-error mitigation plus zero-noise extrapolation).

QUANTUM TECHNOLOGY INITIATIVE

Code Showcase

Examples from the Github code

Applications

pagation of Parametrized Circuits

o.17989023

expectation values of a circuit's observables into explicit functions of the rameters.

e

irst import all the necessary libraries:

```
import pennylane as qml # To define the circuit
from pprop.propagator import Propagator # For Pauli Propagation
```

- Define the circuit as a function of the parameters

```
# Function of parameters
def ansatz(params : list[float]):
    qml.RX(params[0], wires=0)
    qml.RX(params[1], wires=1)

    qml.RY(params[2], wires=0)
    qml.RY(params[3], wires=1)
```

Tutorial

1: Import necessary libraries

```
import pennylane as qml # To define the circuit
from pprop.propagator import Propagator # For Pauli Propagation
```

Tutorial

2: Define the circuit as a Pennylane function of the parameters, put the observables to propagate as return values

```
# Function of parameters
def ansatz(params : list[float]):
    qml.RX(params[0], wires=0); qml.RX(params[1], wires=1)
    qml.RY(params[2], wires=0); qml.RY(params[3], wires=1)
    qml.Hadamard(wires = 2)

    qml.CNOT(wires = [0, 1]); qml.CNOT(wires = [1, 2])

    qml.RY(params[4], wires=0)
    qml.RY(params[5], wires=1)
    qml.RY(params[6], wires=2)

    return [qml.expval(qml.PauliZ(qubit)) for qubit in range(1)] + [qml.exp
```

Tutorial

3: Initialize `Propagator` class

```
prop = Propagator(
    ansatz,
    k1 = None, # Cutoff on the Pauli Weight
    k2 = None, # Cutoff on the frequencies
)
```

Tutorial


```
>>> prop
Propagator
  Number of qubits : 3
  Trainable parameters : 7
  Cutoff 1: None | Cutoff 2: None
  Observables [Z(0), X(0) @ X(1) @ X(2), Y(2), -1.0 * (X(0) @ X(1) @ X(2))
0: ──RX──RY──||─╭●─────||──RY─┤  <Z> ╭<X@X@X>       ╭<ℋ>
1: ──RX──RY──||─╰X─╭●──||──RY─┤      ├<X@X@X>       ├<ℋ>
2: ──H──────||─────╰X──||──RY─┤      ╰<X@X@X>  <Y> ╰<ℋ>
```

Tutorial

4: Propagate

```
prop.propagate()
Propagating -1.0 * (X(0) @ X(1) @ X(2)) + 13.0 * Z(2): 100%|██████████| 4/4
```

Tutorial

Once propagated, we have access to the functional forms of the observables

```
prop.expression()
```

$$Z0 = -sin(\theta_2) * sin(\theta_3) * sin(\theta_4) * cos(\theta_0) * cos(\theta_1) + cos(\theta_0) * cos(\theta_2) * cos(\theta_4)$$

...

5: Which can be evaluated through

```
random_params = qml.numpy.arange(prop.num_params)
prop_output = prop.eval(random_params)
[ 0.32448207 -0.5280619   0.          4.16046337]
```