# Error mitigation using logical shadow tomography

Yanis Le Fur
June 9, 2023 Ryan LaRose

**Abstract**

We present numerical and experimental results obtained for error mitigation techniques on the NISQ computers. We use shadow tomography for state reconstruction and virtual distillation and subspace expansion as error mitigation techniques. In our first experiment, we will be simulating noisy circuit prepared in GHZ-states on virtual machines and quantum computers such as the quantum virtual machine from Google AI and IBM quantum computers. We perform our experiment using five-qubits ibmq-lima quantum computer to have a realistic noise model. In our next experiment we apply those techniques to the $[[5, 1, 3]]$ error correcting code prepared in the logical GHZ-state and simulate noise model using depolarizing noise or a virtual quantum machine.

## 1 Introduction

Noisy Intermediate-Scale Quantum (NISQ) technologies has been of high interest lately due to the large amount of technological progress they could offer for numerous fields such as chemistry [1] or graph problems [2]. However, due to the large error rates in NISQ computers some methods need to be developed in order to reduce the errors and retrieve the correct information. These methods are called Quantum Error Mitigation (QEM) methods and aim to reduce error rate of the quantum system observed. Numerous QEM techniques has been developed such as Variational Quantum Eigensolver (VQE) [3] that use post-processing to directly act on the parameters of the quantum processor (QPU) in order to reduce the error rate. This VQE has direct application to chemistry since it permits modeling bonding energies between molecules [4]. In parallel, recent research in quantum information and quantum computing have shown that it is possible to make prediction of observables using quantum computers and using very few measurements [5]. This technique is named shadow tomography and creates new opportunities for the post-processing error mitigation techniques [6, 7, 9, 14]. In this work, we will use logical shadow tomography which combines two QEM techniques: virtual distillation and subspace expansion. In order to retrieve the final state, we will use shadow tomography with single qubit Pauli gates. We will study error mitigation on quantum circuits with different error rates using virtual and physical machines. Finally, we will implement the $[[5, 1, 3]]$ code and perform logical shadow distillation with a logical Bell state.

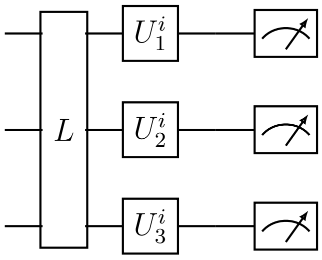
## 2 Background

In order to perform the measurement of an observable value, one way is to reconstruct the density matrix from the output state $\rho = |\psi\rangle\langle\psi|$ and we get that $\langle O\rangle = \mathrm{Tr}(\rho O)$. However, while performing the calculation the computer is noisy. Thus, we get the noisy density matrix $\rho_{\mathcal{E}} = \mathcal{E}(\rho)$ and we evaluate $\langle O\rangle_{noisy}$. Our goal is thus using QEM techniques to find $\langle O\rangle_{QEM}$ such that:

$$\left|\langle O\rangle - \langle O\rangle_{QEM}\right| < \left|\langle O\rangle - \langle O\rangle_{noisy}\right|.$$

### 2.1 Shadow Tomography

The first step of error mitigation is to retrieve the output state of our circuit and for that, we will use logical shadow tomography technique [5, 6, 10]. We can decompose this technique in two steps: First, we perform several measurement around the whole Bloch sphere of our space by appending a random unitary transformation $\mathcal{U}$ to the circuit . In our case, we will use Pauli measurement meaning the gate will be single qubit gates and are thus given by $\{HS, H, I\}$. We give an example of a 3-qubits circuit with random measurement schedule:



where the gate $L$ represent any operation through the circuit. The circuit is repeated $N_s$ times with $i = 1, 2...N_s$. From the measurement schedule performed using $\mathcal{U}$ we get a set of bit string $b = \{|b\rangle : b \in \{0,1\}^N\}$. Now that we have a set of measurement record from unitary random transformation we need to reconstruct the noisy density matrix $\rho_{\mathcal{E}}$. For that we construct the classical shadow ensemble:

$$\rho_s = \left\{ \mathcal{M}_N^{-1} \left( \bigotimes_{i=1}^{N} U_i^s |b_i^s\rangle \langle b_i^s| (U_i^s)^\dagger \right) \right\}_{s=1}^{N_s} \tag{1}$$

where $\mathcal{M}_N^{-1}$ is the classical shadow reconstruction map that is dependant of $\mathcal{U}$ and is unique [5].

To prove the uniqueness, we need to assume that the set $b$ is tomographically independent meaning that for each $\rho \neq \sigma$ there exist $U \in \mathcal{U}$ and $|b\rangle \in b$ such that $\langle b| U\sigma U^\dagger |b\rangle \neq \langle b| U\rho U^\dagger |b\rangle$. Now, we also have that during a random measurement we perform the operation $\rho_s \to U^\dagger |b\rangle \langle b|$. We thus get by Born's Rule:

$$P(\hat{b} = b) = \langle b| U\rho U^\dagger |b\rangle \tag{2}$$

Then, we get that the reconstruction map is given by:

$$\mathcal{M}(\rho) = \mathbb{E}_{U \in \mathcal{U}} \sum_b \langle b| U\rho U^\dagger |b\rangle U^\dagger |b\rangle \langle b| \tag{3}$$

Thanks to the tomography independence we can ensure that the inverse reconstruction map is unique and that

$$\rho = \mathbb{E}_s \mathcal{M}_N^{-1} \left( \bigotimes_{i=1}^{N} U_i^s |b_i^s\rangle \langle b_i^s| (U_i^s)^\dagger \right). \tag{4}$$

We also find that:

$$\mathcal{M}_N^{-1} = (2^N + 1)\rho - \mathbb{I}. \tag{5}$$

Now, by using single qubit pauli matrices as unitary transformation we have that the reconstruction matrix is given by $\mathcal{M}_N^{-1} = \bigotimes_{i=1}^{N} \mathcal{M}_1^{-1}$ [10]. Finally, we get that the reconstructed density matrix by shadow tomography is given by:

$$\rho = \mathbb{E}_s \bigotimes_{i=1}^{N} \mathcal{M}_1^{-1} \left( U_i^s |b_i^s\rangle \langle b_i^s| (U_i^s)^\dagger \right). \tag{6}$$

## 2.2 Shadow Distillation

Now that we have reconstructed the density matrix $\rho_{\mathcal{E}}$ using shadow tomography we still need to mitigate the errors. For that we have two methods at our disposition. The first one is called virtual distillation and makes use of powers of the density matrix. The second one is called subspace expansion and make use of the stabilizers from error correction.

### 2.2.1 Virtual distillation

This method reside in taking the power of the noisy density matrix. One can see that taking the power of density matrix reduce the noise in our system simply by taking the thermal density matrix $\rho = e^{\beta H}$ where $\beta = (k_B T)^{-1}$ and considering that the temperature $T$ is the noise source. Then by taking the power of the density matrix $\rho^M = e^{M\beta H}$ we divide the temperature by $M$ and then reduce the noise.
We can prove this rigorously by assuming that the noiseless state has the higher eigenvalue (or is dominant compared to the other states) [12]. Indeed, by assuming that $\rho_{\mathcal{E}} = p_0 |\psi_0\rangle \langle\psi_0| + \sum_k p_k |\psi_k\rangle \langle\psi_k|$, we also define that $\langle\psi_i|\psi_j\rangle = \delta_{i,j}$ and that the probabilities are ordered such that $p_0 > p_1 \geq p_2 \geq p_3 \geq ... \geq 0$. By calculating the average value of the observable using the power $M$ of the density matrix we get that:

$$\begin{aligned}\langle O\rangle_{VD} = \frac{\mathrm{Tr}\left(\rho^M O\right)}{\mathrm{Tr}\left(\rho^M\right)} &= \frac{p_0^M \langle\psi_0| O |\psi_0\rangle + \sum_{k=1} p_k^M \langle\psi_k| O |\psi_k\rangle}{p_0^M + \sum_{k=1} p_k^M} \\ &= \langle\psi_0| O |\psi_0\rangle \left[\frac{1 + \sum_{k=1}(p_k/p_0)^M \langle\psi_k| O |\psi_k\rangle / (\langle\psi_0| O |\psi_0\rangle - 1)}{1 + \sum_{k=1}(p_k/p_0)^M}\right] \\ &= \langle\psi_0| O |\psi_0\rangle \left[1 + \mathcal{O}(p_1/p_0)^M)\right]\end{aligned} \tag{7}$$

We thus observe that by taking $M \to \infty$ we retrieve the noiseless average value of the observable $O$.

### 2.2.2 Subspace expansion

The subspace expansion method reside in the use of the stabilizers from the error correcting code (more informations about error correcting code in [13]) in order to remove the correctable errors detected by the stabilizer group. Indeed, for an stabilizer group $\mathcal{S} = \langle S_1, ..., S_n\rangle$ we have that for $|\psi\rangle$ generated by logical operation: $\forall i, S_i |\psi\rangle = |\psi\rangle$ and for an correctable error $E_i$, $\exists S_i$ s.t. $S_i E_i |\psi\rangle = -|\psi\rangle$. By using the anticommutating relation between the stabilizer group and the correctable error group we can generate projectors $P_i$ that will map any $\hat{\psi}$ to the purified $\psi_0$ (see Fig.1). We define the projectors by:

$$P_i = \frac{\mathbb{I} + S_i}{2} \tag{8}$$

and we define the full projection operator $\Pi$ by:

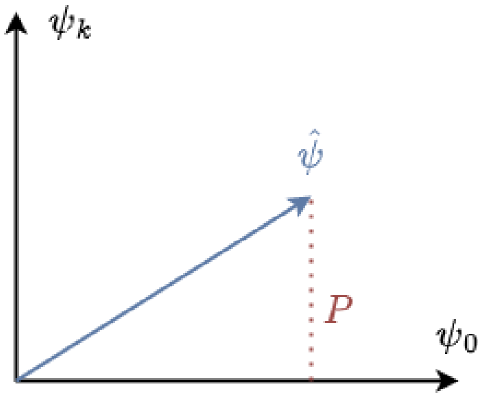


Figure 1: Representation of the action of a projector $P$ on a state $\hat{\psi}$ on the space of the purified state $\psi_0$ and the noisy state $\psi_k$.

$$\Pi = \prod_{i=1}^{n} P_i = \prod_{i=1}^{n} \frac{\mathbb{I} + S_i}{2}. \tag{9}$$

Now that we have define the projection operator $\Pi$, we can define the purified average value of the observable $\langle O\rangle_P$. We define the decomposition of $O$ by pauli operator by $O = \hat{\gamma}\sum_j \gamma_j O_j$ where $O_j$ is a pauli operator. Now, by using the cyclic symmetry of the trace and the definition of the projector such that $\Pi^2 = \mathbb{I}$ we get that [14]:

$$\begin{aligned}\langle O\rangle_P = \mathrm{Tr}\left(\Pi\rho\Pi O\right) &= \frac{\hat{\gamma}}{2^n}\sum_{i=1}^{n}\gamma_i \mathrm{Tr}\left(\rho O_i(\mathbb{I}+S_i)\right) \\ &= \frac{\hat{\gamma}}{2^n}\sum_{\chi\in\{0,1\}^n}\sum_{i=1}^{n}\gamma_i O_{\chi,i}\end{aligned} \tag{10}$$

with $O_{\chi,i} = \mathrm{Tr}\left(\rho O_i S_\chi\right)$ for $S_\chi = S_1^{\chi_1}...S_n^{\chi_n}$.

Now that we defined the virtual distillation and the subspace expansion method we can combine them in order to mitigate the noise in our system. We thus get that our mitigated average value is given by:

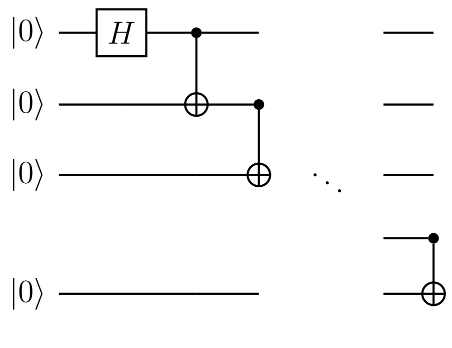
$$\langle O\rangle_{QEM} = \frac{\mathrm{Tr}(\Pi\rho^M O\Pi)}{\mathrm{Tr}(\rho^M)}. \tag{11}$$

## 2.3 Greenberger–Horne–Zeilinger state

GHZ-states are perfectly entangled states such that for $N$ qubits we have that

$$|\mathrm{GHZ}\rangle = \frac{|0\rangle^{\otimes N} + |1\rangle^{\otimes N}}{\sqrt{2}}.$$

For the case where $N = 2$ this is the Bell state $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. These states are frequently used for quantum metrology and quantum information since they permit an enhancement of the sensitivity of the measurement compared to non-untangled states [15]. The preparation of these states only required hadamard and controlled-X gates and can be generalized for any number of qubits $N$. This can be prepared using the following circuit:



## 2.4 $[[5,1,3]]$ error correction code

In order to apply both of our mitigation error technique, we will perform shadow distillation on circuits encoded with the $[[5,1,3]]$ error correcting code. Our application will be using two logical qubits encode with this error correcting code. For that we need to review how to encode the five qubit code, how to perform logical operation such as hadamard or CNOT but also define the logical operator and stabilizer of the circuit.

### 2.4.1 Stabilizers and Logical operators

We can define the $[[5, 1, 3]]$ error correcting code with different stabilizer group. In our case, we define the stabilizer group $\mathcal{S}_1 = \langle S_1, S_2, S_3, S_4 \rangle$ of this code by:

$$\begin{aligned} S_1 &= XZZXI \\ S_2 &= IXZZX \\ S_3 &= XIXZZ \\ S_4 &= ZXIXZ \end{aligned} \tag{12}$$

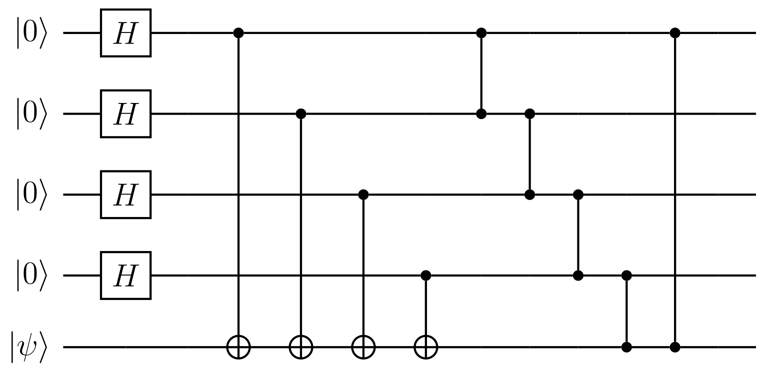
where the logical operator are given by $X_L = XXXXX$ and $Z_L = ZZZZZ$. Now, we create the $[[10, 2, 3]]$ code by simply concatenating two $[[5, 1, 3]]$ code thus the stabilizers and logical operators can be derived from the $[[5, 1, 3]]$ code. Indeed the stabilizers are given by

$$\mathcal{S}_2 = \langle A \otimes B | A, B \in \mathcal{S}_1 \rangle$$

and the logical operators become $X_L I_L$, $I_L X_L$,$Z_L I_L$ and $I_L Z_L$.

### 2.4.2 Encoding

In this section, we aim to encode any state $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ into the logical state equivalent for the five qubits code defined by the stabilizers in (12). A way of encoding the five qubits code is by performing the circuit from [16]:

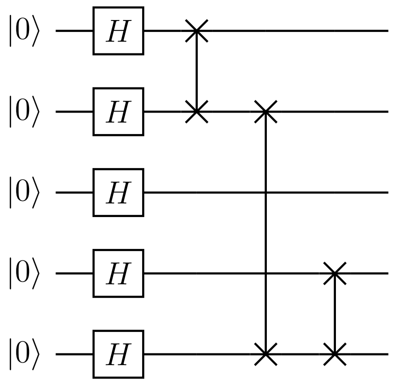


In this way we get the encoding $|\psi\rangle \rightarrow |\bar{\psi}\rangle = a\,|\bar{0}\rangle + b\,|\bar{1}\rangle$.

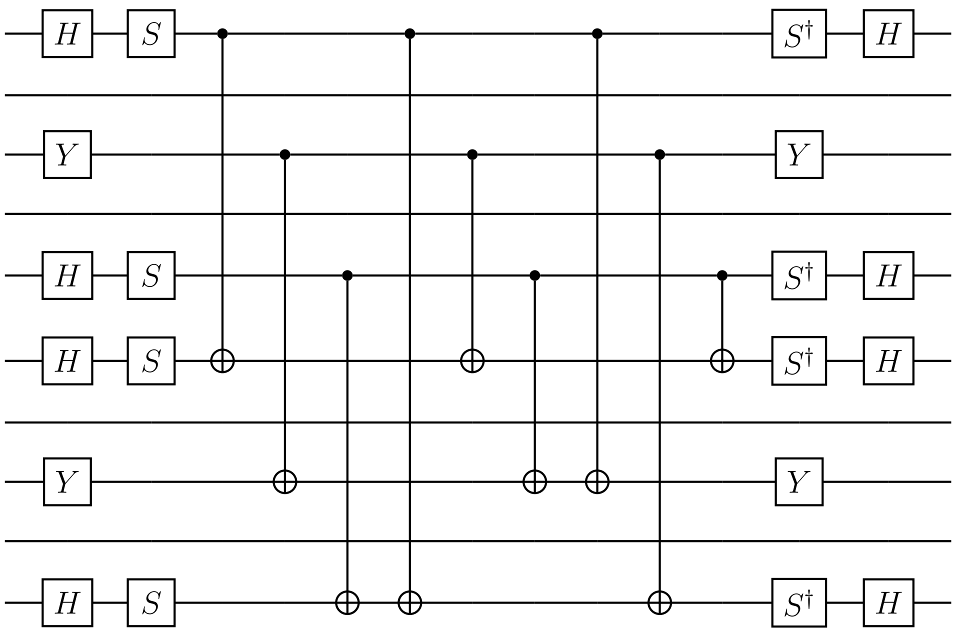
### 2.4.3 Logical operation

Now that we encoded our logical qubit we would like to create a GHZ state from our two $[[5, 1, 3]]$ codes (or on the $[[10, 2, 3]]$ code). For that we need to define the logical hadamard gate and the logical CNOT gate.

**Logical hadamard:** The logical hadamard gate can be perform by using single qubit hadamard gates and SWAP gates. We can use the circuit from [17] which is given by:



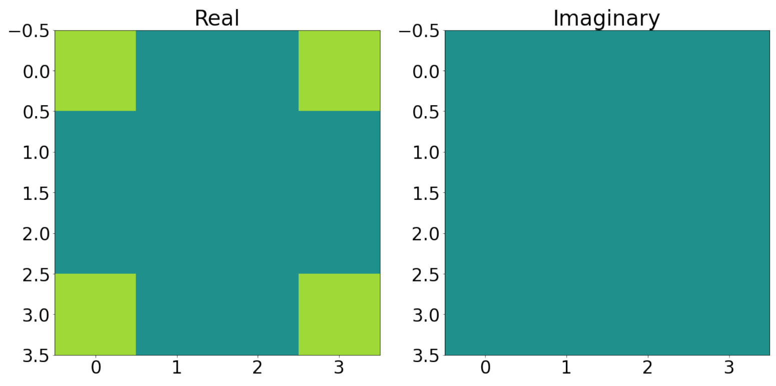
**Logical CNOT:** The $[[5,1,3]]$ code does not admit transversal CNOT gate. This mean we cannot simply apply CNOT gates to each individual qubits in order to perform the logical CNOT gate. However, by applying a rotation in our system given by $U = H_1S_1Y_3H_5S_5$ and $V = H_6S_6Y_8H_{10}S_{10}$ and a serie of CNOT gate it is possible to create a logical CNOT gate [18].This circuit will leave the stabilizers of the 10 qubits code unchanged while performing the corresponding operation on the logical operators: $X_LI_L \to X_LX_L$, $I_LX_L \to I_LX_L$, $Z_LI_L \to Z_LI_L$ and $I_LZ_L \to Z_LI_L$. This logical CNOT gate can be performed using the following circuit:



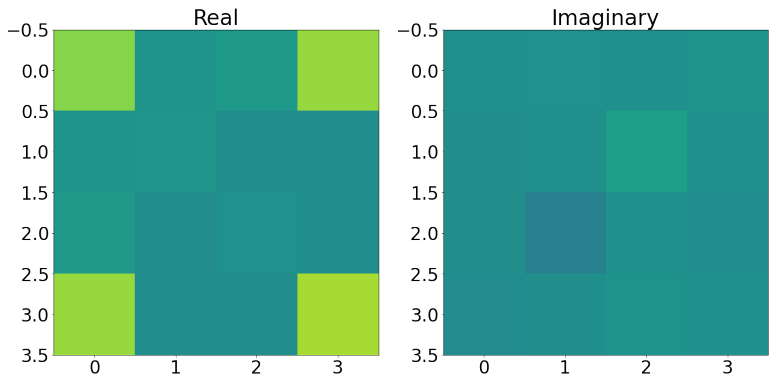
## 3 Results and Discussion

### 3.1 Simple two qubit experiment

In this section, we will present the results obtained using shadow tomography and virtual distillation for 2-qubits circuit using Cirq [21]. For that we will prepare our two qubits in the Bell state $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Now since shadow tomography aim to reconstruct density matrix, we will work with the density matrix $\rho = |\Phi^+\rangle\langle\Phi^+|$. We first present the actual density matrix of our circuit $\rho$ compared to the reconstructed one using shadow tomography $\rho_s$ for a noiseless circuit in Fig.2.



(a)



(b)

Figure 2: Representation for the real and imaginary value of (a) the actual density matrix $\rho$ and (b) the reconstructed density matrix $\rho_s$ via shadow tomography for $n_s = 10^3$ shots.

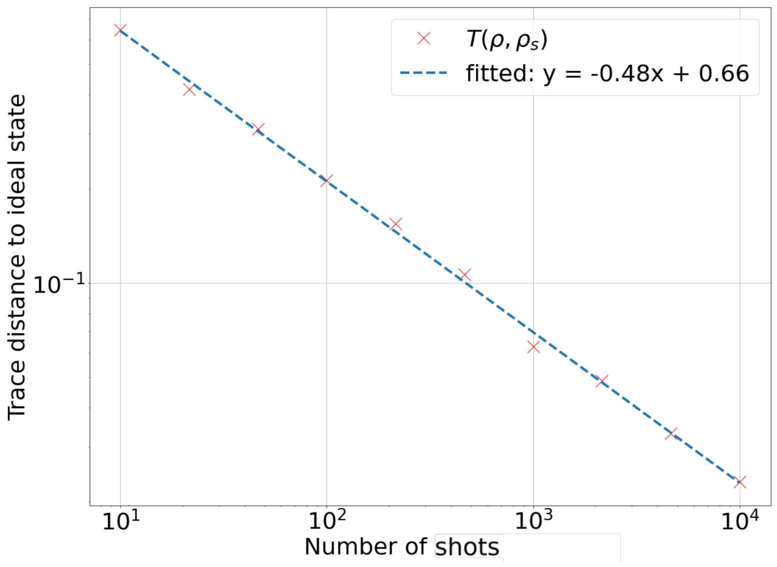


Figure 3: Trace distance $T(\rho, \rho_s)$ between the actual density matrix $\rho$ and the reconstructed density matrix $\rho_s$ as a function of the number of shots $n_s \in [10^1, 10^4]$.

We see that for $n_s = 10^3$ number of shots we can reconstruct pretty accurately the density matrix $\rho$ using shadow tomography. Now we would like to know how does that evolve with the number of shots $n_s$. To measure the accuracy, we will use the trace distance between the actual density matrix $\rho$ and the reconstructed one $\rho_s$. We thus get the trace distance as a function of the number of shots in Fig.3. The trace distance starts at order 0.5 and hits a precision under $10^{-1}$ at $n_s = 10^3$. Now that we performed shadow tomography for a noiseless circuit, we will be interested into noisy circuit in order to perform error mitigation. We will study the evolution of the trace distance for the same circuit with a depolarizing noise at each gate where we will be varying the depolarizing noise rate. Due to the small amount of qubits, the simulation time is relatively low and permit to perform shadow tomography at high number of shots. Here we will be taking $n_s = 10^4$. We present the trace distance for depolarizing probabilities $p \in [0, 0.6]$ in Fig.4.

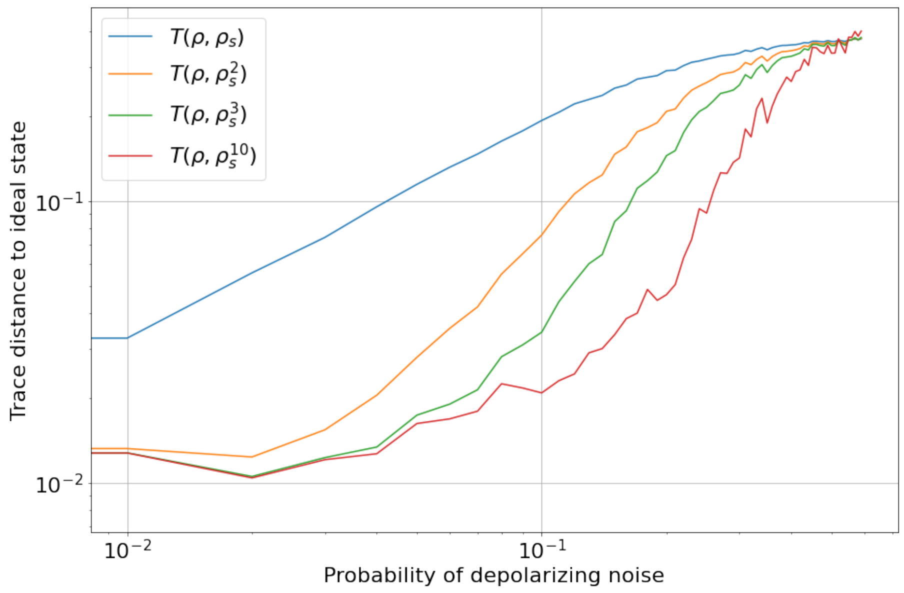


Figure 4: Evolution of the trace distance $T(\rho, \rho_s^n)$ between the actual density matrix $\rho$ and the reconstructed and trace normalized density matrix $\rho_s$ taken at the power $n$ for $n_s = 10^4$ number of shots and depolarizing noise $p \in [0, 0.6]$ for a 2−qubit code.

We performed virtual distillation on the reconstructed density matrix $\rho_s$ on Fig.4. We observe that performing virtual distillation at higher order reduce the trace distance but also delay the phase transition between the low noise state and the totally noisy state which is in accord with [11].

As a last 2-qubits experiment, we will be computing the values of $\langle X_1 X_2 \rangle$ and $\langle Z_1 Z_2 \rangle$ by peforming shadow tomography and virtual distillation. Theoretically, the values for those observable with a GHZ state should be both equal to one. We present the results for a depolarizing noise probability of 0.02 in Fig.5. We remark that the average value of the observables using only shadow tomography is approximately 0.8 and using virtual distillation with very low power $m = 3$ we achieve getting close average values to the theoretical average value.

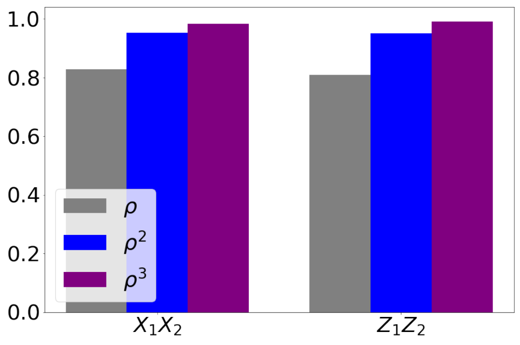


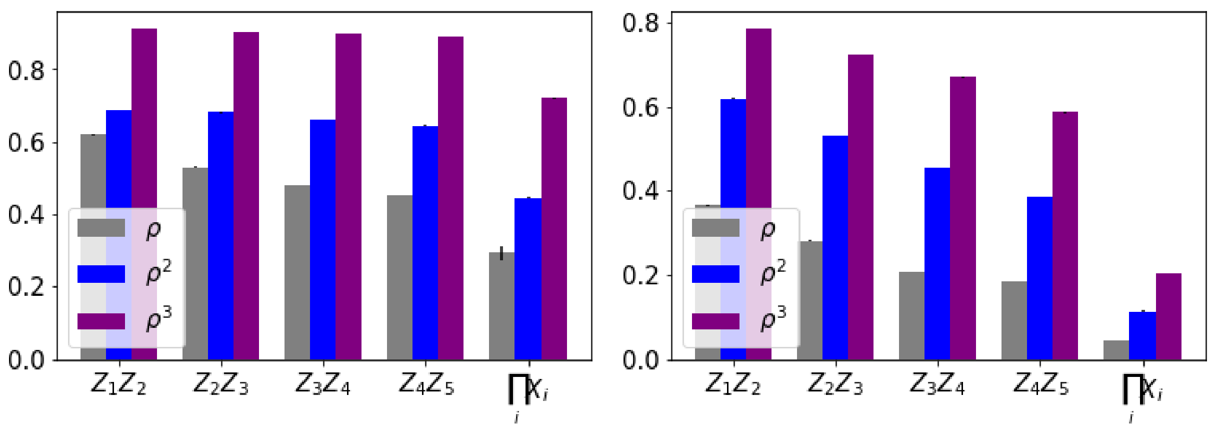
Figure 5: Comparison of the value of $\langle O \rangle$ for different observables using virtual distillation and shadow tomography to reconstruct a 2-qubit GHZ state.

### 3.2 five-qubit experiment

After presenting the utility of the virtual distillation method for 2-qubits circuit, we will perform our scheme for 5-qubits circuit. We follow the experiment in [7] that use trapped-ion computer in order to demonstrate the efficiency of virtual distillation. In order to simulate our circuit we will be using different supports : Cirq [21], a Quantum virtual machine from Google QVM [24] and IBM quantum computers [25]. For all of these simulators, we will generate 5-qubits noisy circuit and measurement the average value of the observable $O \in \{Z_1Z_2, Z_2Z_3, Z_3Z_4, Z_4Z_5, \prod_i X_i\}$. The noiseless value is $\langle O \rangle = 1$ for every observable. We will measure the average value of the observables using $\langle O \rangle = \frac{Tr(\rho O)}{Tr(\rho)}$. We will retrieve the density matrix $\rho_s$ using shadow tomography and apply error mitigation using virtual distillation.

### 3.2.1 Noisy simulation

We start the trapped Ion experiment using Cirq simulator. We use the same parameters than [7] which are the number of different rotation $N_U = 1500$ and the number of shots for each configuration being $N_S = 50$. We choose to present the average value of the observables for depolarizing noise probability being $p = 0.05$ and $p = 0.1$ where we perform $N_S = 500$. We repeat this experiment $n_{ave} = 4$ times in order to take the mean and variance of $\langle O \rangle$. We present the result obtained for the average value using shadow tomography and virtual distillation in Fig.6.



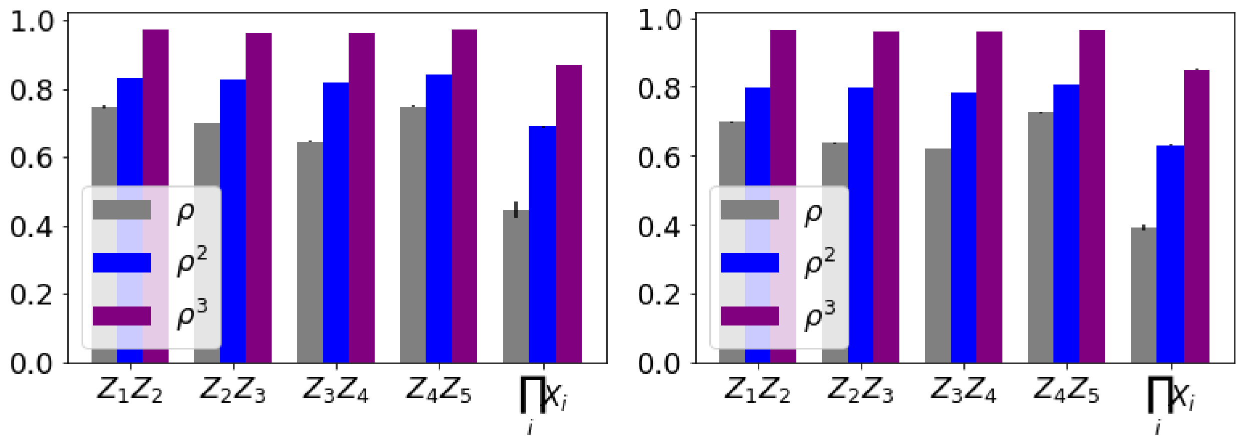
(a) (b)

Figure 6: Comparison of the average value of an observable $O$ measured with the reconstructed $\rho_s$ using shadow tomography and using virtual distillation for power $m = 2, 3$ for depolarizing probability (a) $p = 0.05$ and (b) $p = 0.1$ for a $5-$qubit code.

We remark in Fig.6 that despite the low value for $\langle O \rangle$ using only shadow tomography, virtual distillation permit retrieving the average value with a good accuracy. Another important element to note is the difficulty our model has to reconstruct the observable $\prod_i X_i$. This was expected since reconstructing a average value of an observable with more qubits it's exponentially harder [5]. We also note that virtual distillation do not permit retrieving the state due to the very low value obtained (up to 0.6 for $p = 0.05$ in Fig.6.a and up to 0.25 for $p = 0.1$ in Fig.6.b).

### 3.2.2 Google quantum virtual machine

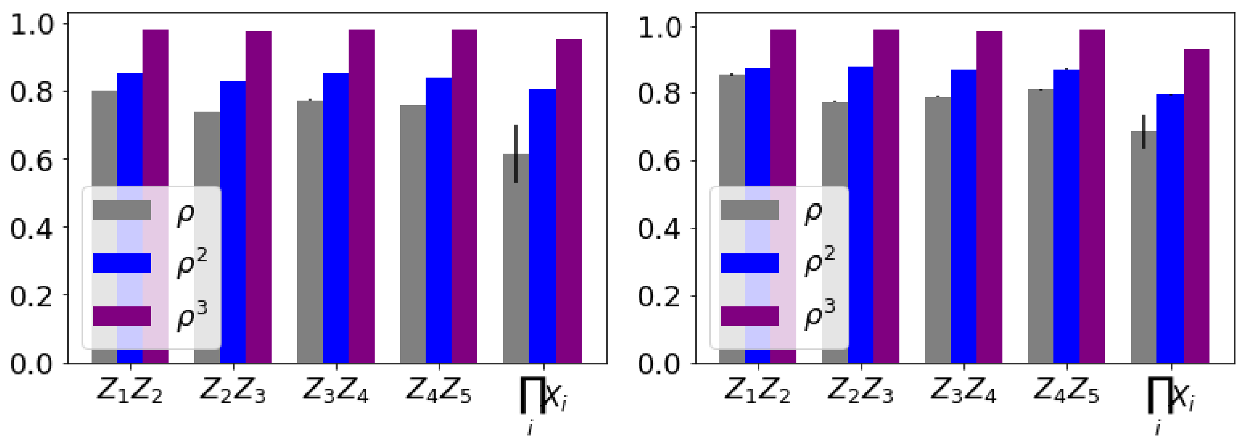
We now present the trapped ion experiment using the quantum virtual machine from Google [24]. We will be using two different processor: Weber (with 53 qubits) and Rainbow (23 qubits). We will also translate our quantum circuit to a more optimized one for the quantum virtual machine that use one of those two-qubits gate: ISWAP or Sycamore. We perform this experiment with the same number of rotation $N_U = 1500$, number of shots $N_S = 50$ and number of repetition $n_{ave} = 4$. We present the results obtained using weber processor using those parameters in Fig.7.



(a) (b)

Figure 7: Comparison of the value of $\langle O \rangle$ for different observables, using shadow tomography and virtual distillation for the quantum virtual machine with weber processor (a) using ISWAP gates and (b) using Sycamore gates for a 5−qubit code.

We also present the results using rainbow processor and the same parameters in Fig.8.



(a) (b)

Figure 8: Comparison of the value of $\langle O \rangle$ for different observables, using shadow tomography and virtual distillation for the quantum virtual machine with rainbow processor (a) using ISWAP gates and (b) using Sycamore gates for a 5−qubit code.

Compared to Fig.6, we remark that the results obtained using the QVM in Fig.7 or Fig.8 give higher values for $\langle O \rangle$ for any observable $O$. We can deduce that the simple depolarizing noise model used in Fig.6 was not representing our system correctly. Now, to compare the different use of gates, there is no significant advantage using Sycamore or ISWAP gate as seen in Fig.7.a and Fig.7.b or in Fig.8.a and Fig.8.b. We denote however, a slight improvement for the observable $\prod_i X_i$ using Sycamore gates for the rainbow processor even though it is accompanied with an increase of the variance. Finally, using virtual distillation permit getting closer values to the theoretical value of $\langle O \rangle$. The best use of virtual distillation is seen in Fig.7.a where we get value such as 0.4 for $\prod_i X_i$ using simple shadow tomography being increased up to 0.9 using only third power virtual distillation.

### 3.2.3 IBM quantum computer

Now that we performed shadow tomography and virtual distillation using Cirq and the quantum virtual machine, we propose to run our code on the IBM quantum computers [25]. In order to respect the conditions imposed by IBM and reduce the simulation time, we will choose to perform shadow tomography with all the different configuration and a total of simulation $n_s = 75000$. We will perform our simulation using the quantum computer ibmq-lima and we present the results obtained in Fig.9. We remark the same tendancy of getting high average values for the two pauli observables while having lower average value for the $\prod_i X_i$ observable which is in respect with the complexity of reconstructing higher dimensional observables.

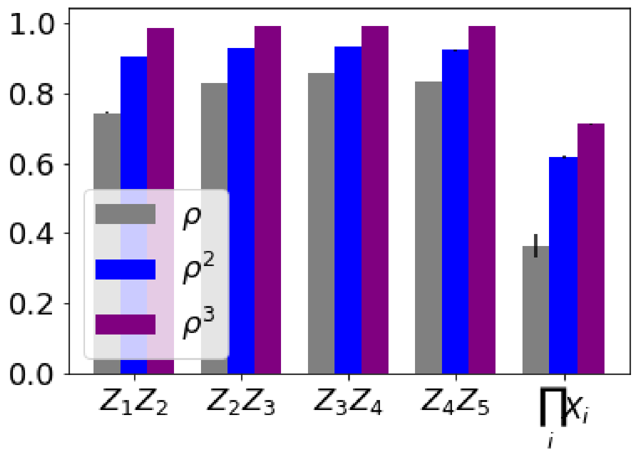


Figure 9: Comparison of the value of $\langle O \rangle$ for different observables, using shadow tomography and virtual distillation for the IBM quantum computer ibmq-lima for a 5−qubit code.

## 3.3 Two qubits encoded in the $[[5, 1, 3]]$ code

We perform an implementation using Cirq of two logical qubits code encoded in the $[[5, 1, 3]]$ code as described in Sec.2.4. We perform shadow tomography as a state reconstruction technique followed by error mitigation technique: virtual distillation and subspace expansion. We will perform simulation using Cirq and QVM from Google AI. We will be interested in the average values of the $X_LX_L = \prod_{i=1}^{10} X_i$ and $Z_LZ_L = \prod_{i=1}^{10} Z_i$ observables. Those observables have the following average values for the GHZ state encoded in the $[[10, 2, 3]]$error correction code: $\langle X_LX_L \rangle = 1$ and $\langle Z_LZ_L \rangle = 1$. We start by presenting the evolution of the average values as a function of the power take in order to perform virtual distillation using or not subspace expansion and using shadow tomography for $n_s = 4 \cdot 10^7$ number of shots in Fig.10.

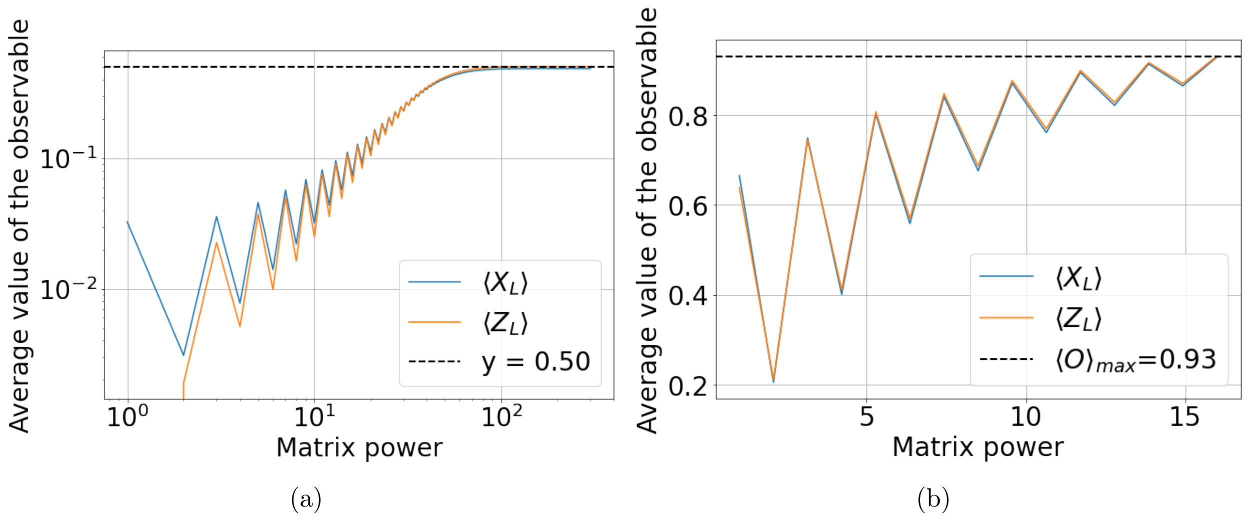


Figure 10: Evolution of the average values of the $X_LX_L$ and $Z_LZ_L$ using (a) virtual distillation and (b) virtual distillation combined with subspace expansion with cirq depolarizing noise simulator for two logical qubit circuit encodode in the $[[5, 1, 3]]$ code.

|  | non-purified | purified |
|---|---|---|
| $\langle X_L X_L \rangle$ | 0.49 | 1.00 |
| $\langle Z_L Z_L \rangle$ | 0.50 | 1.00 |

Table 1: Average values of the observables $X_L X_L$ and $Z_L Z_L$ using virtual distillation at infinite power and purifying the state or not using subspace expansion for the 2 logical qubits encoded in the $[[5, 1, 3]]$ code with depolarizing noise simulation.

We observe from Fig.10.a and Fig.10.b that subspace expansion gives an advantage in terms of error mitigation capability. Indeed, using only virtual distillation does not permit getting average values close to one even using density matrices taken at high power. Even at infinite power we get that the average values are $\langle X_L X_L \rangle = 0.48$ and $\langle Z_L Z_L \rangle = 0.5$. We thus get a clear asymptote at 0.5. This asymptote might be created due to the noisy encoding of the logical state. Indeed, for more realism, we encoded the logical state with a noisy circuit thus, this have for effect of not creating the exact logical state and thus giving asymptote behaviour for the average value of the observables.

When performing subspace expansion coupled to virtual distillation we get values over 0.8 only at power $m = 7$ and close value to 1 at $m = 15$. It seems that the subspace expansion permit retrieving the correct average values even with the noisy encoding.

In order to grasp the incapacity of retrieving the correct density matrix using only virtual distillation, we represent the sorted eigenvalue distribution before and after subspace expansion in Fig.11.

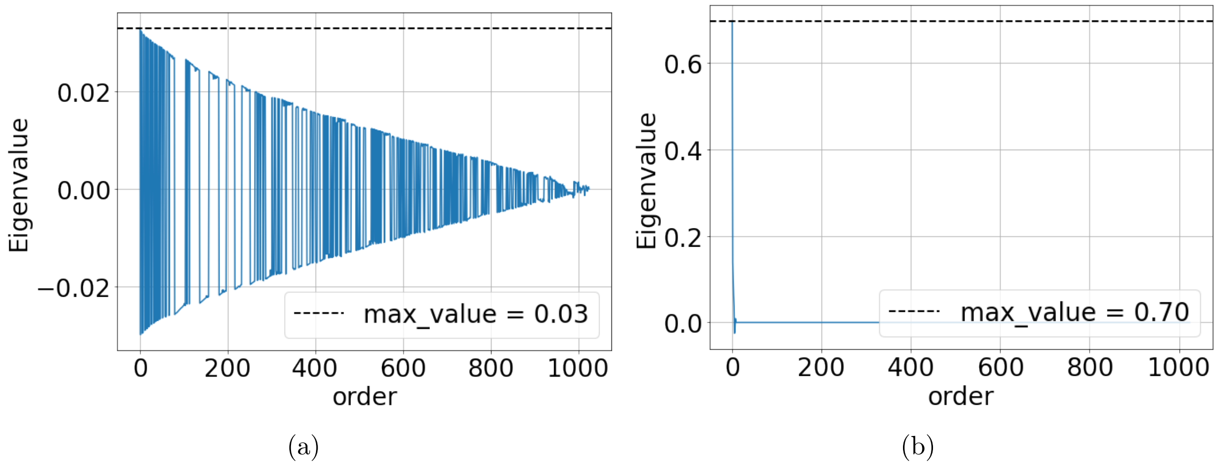
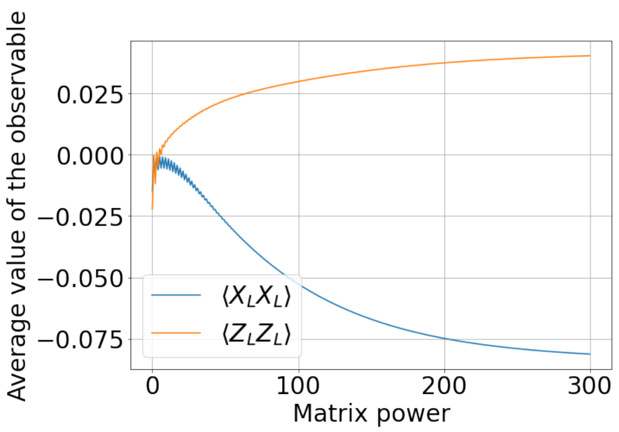


Figure 11: Sorted eigenvalue distribution of the reconstructed density matrix (a) using shadow tomography (b) using shadow tomography and subspace expansion using Cirq simulator.

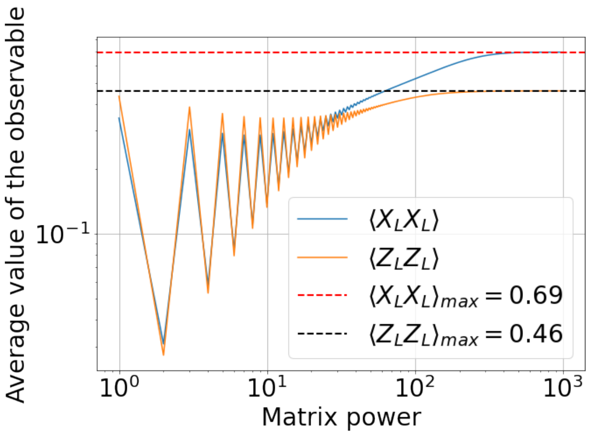
One should note from Fig.11 that we are getting negative eigenvalues meaning that the density matrix we are retrieving is highly nonphysical. This is expected since shadow tomography or tomography in general will reconstruct nonphysical states [19, 20]. To continue on the average value of the observable, it is clear that due to the density of eigenvalues with close values in Fig.11.a compared to Fig.11.b, it will be harder to reconstruct the real state using only virtual distillation. Indeed, let's take the two first eigenvalues from Fig.11.a which are given by: $\lambda_1 = 3.3 \cdot 10^{-2}$ and $\lambda_2 = 3.1 \cdot 10^{-2}$. It seems clear that we will need to perform virtual distillation at high power since $p_0$ and $p_1$ from (7) will be really close. On the other

hand, performing virtual distillation after subspace we be much easier since the first eigenvalue is of higher order than the other ones (Fig.11.b). The study of eigenvalues can also help us understand the unstable evolution of the average value using shadow distillation in Fig.10. Indeed, we observe that the average values are smaller for power $m = 2k$ than $m = 2k + 1$ for $k \in \mathbb{N}$. This is caused by the negative eigenvalues of the reconstructed density matrix taken to the square becoming positive and being close the the real state eigenvalue. Indeed for $m = 2$ we get that the two first eigenvalues are given by: $\lambda_1 = 1.1 \cdot 10^{-3}$ and $\lambda_2 = 1.0 \cdot 10^{-3}$. This closeness can be reduce if one increase the number of shots performed to recover the density matrix. By doing so, it will increase the gap between the real state eigenvalue and the noisy state eigenvalues.
We now are interested in the average values of the same observables using here the quantum virtual machine from Google AI. We will be performing shadow tomography with the same number of shots $n_s = 4 \cdot 10^7$. We present the evolution of the average value of the observables using virtual distillation alone or coupled with subspace expansion in Fig.12. It is interesting to find that the virtual distillation does not permit at all to recover the correct eigenvalues even at high power (see Fig.12.a). Whereas using subspace expansion permit retrieving higher values for the average of the observables, it seems that we are having the same asymptote problem than when using the depolarizing noise with virtual distillation only. We can observe this asymptote with the value of the observables at infinite power in Tab.2.

To explain the incapacity of retrieving high average value for the observables using only virtual distillation, we present the eigenvalues of the density matrix in Fig.13. As seen for the Cirq simulation, doing subspace expansion permit reducing all the eigenvalues except the one from the true state as seen in Fig.13. We can see that from the closeness between the eigenvalues from Fig.13.a. Indeed the first two eigenvalues are given by $\lambda_1 = 3.29 \cdot 10^{-2}$ and $\lambda_2 = 3.26 \cdot 10^{-2}$. .

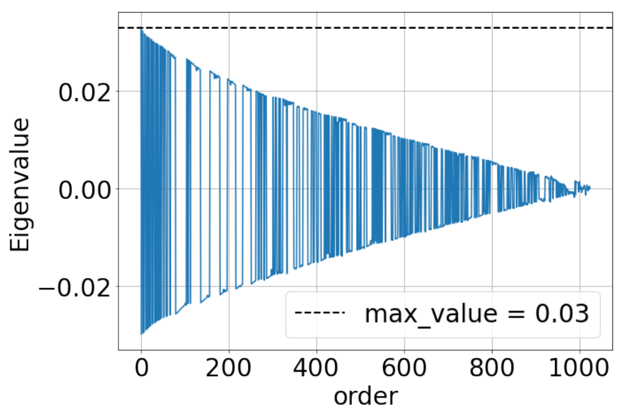


(a)

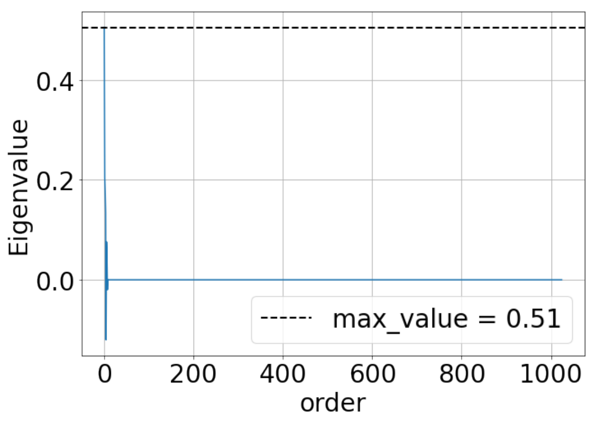


(b)

Figure 12: Evolution of the average values of the $X_L X_L$ and $Z_L Z_L$ using (a) virtual distillation and (b) virtual distillation combined with subspace expansion for the 2 logical qubits encoded in the $[[5, 1, 3]]$ code with the quantum virtual machine simulator.



(a)



(b)

Figure 13: Sorted eigenvalue distribution of the reconstructed density matrix (a) using shadow tomography (b) using shadow tomography and subspace expansion using QVM simulator.

| | non-purified | purified |
|---|---|---|
| $\langle X_L X_L \rangle$ | $-0.08$ | 0.90 |
| $\langle Z_L Z_L \rangle$ | 0.04 | 0.77 |

Table 2: Average values of the observables $X_L X_L$ and $Z_L Z_L$ using virtual distillation at infinite power and purifying the state or not using subspace expansion for the 10-qubits QVM simulation.

## 4 Method

### 4.1 Cirq

In our research, we will be using Cirq [21], an open source framework for programming quantum computers available on Python. We will be using it in order to create our quantum circuit, measuring and perform shadow tomography to reconstruct our denstiy matrix[1].

### 4.2 Shadow tomography using Cirq

In order to perform shadow tomography, we first need to prepare our state. In our case we prepare it to $N$-qubits GHZ-state and this is given by:

```
qubits = cirq.LineQubit.range(N)
cirq.Circuit(cirq.H(qubits[0]),
            *[cirq.CNOT(qubits[i - 1], qubits[i]) for i in range(1,len(qubits))],
            )
```

Once we get our circuit prepared, we need to perform random single qubit rotations and measure the outcome values. We first define a scheme of measurement in each basis ($X$,$Y$ and $Z$) for each of the $N$ qubits. Then, we take all of the rotation possibilities in the scheme and count them. Now, we go through all the rotations and apply the corresponding gates to the measurement

[1]code is available at: https://github.com/YanisLeFur/Logical-Shadow-Tomography

basis. Thus, for measuring in the $X$ basis we apply an hadamard gate for measuring in the $Y$ basis we apply hadamard following by a phase gate $S$. For the $Z$ measurement basis no gate are needed since the measurement are defined to be done in the $Z$ basis. We perform this gate application by defining the following function:

```
def bitGateMap(c,g,qi,qubits)
    if g=="X":
            c.append(cirq.H(qubits[qi]))

    elif g=="Y":
        sdg = cirq.S**-1
        c.append(sdg(qubits[qi]))
        c.append(cirq.H(qubits[qi]))

    elif g=="Z":
        pass
    else:
        raise NotImplementedError(f"Unknown gate {g}")
```

This function take as an argument the Cirq circuit, a string being the measurement basis, the index of the qubit to the corresponding measurement basis and the qubits of the system. It return the circuit with the added corresponding gate from the measurement basis. We then perform our measurement using the following function:

```
    def measure(c,qubits):
        c.append(cirq.measure(*qubits, key='result'))
        return c
```

This add a measurement operation to each qubits with a key named `'result'`. Now that our circuit is prepared in a measurement basis we simulate it using qsimcirq [22], run virtually our circuit for a number of repetition `count` and generate a dictionary named `counts` of our measurement outcomes. Finally, we append the measurement outcomes `counts` of one of the rotation to the list of measurement outcomes named `results`. The code used for the generation of `results` is given by:

```
scheme = [rng.choice(['X','Y','Z'],size=N) for _ in range(nsimu)]
labels, counts = np.unique(scheme,axis=0,return_counts=True)
results = []
for bit_string,count in zip(labels,counts):
    c_m = circuit.copy()
    for i,bit in enumerate(bit_string):  bitGateMap(c_m,bit,i,qubits)
    measure(c_m,qubits)
    if probability>0.:
        c_m = c_m.with_noise(cirq.depolarize(p=probability))
    s = qsimcirq.QSimSimulator()
    samples = s.run(c_m, repetitions=count)
    counts = samples.histogram(key='result')
    results.append(counts)
```

One can note that we add simple depolarization noise with probability `probability` by adding or not the line `c_m = c_m.with_noise(cirq.depolarize(p=probability))`. If one want to control the number of gate `N_U` and the number of simulation for each rotation configuration `N_S`, one can replace nsimu by N_U and only loop through scheme and not labels and counts and finally replace s.run(c_m, repetitions=count) by s.run(c_m, repetitions=N_S). Now that we created our measurement outcomes we need to reconstruct our density matrix rho_shadow. Due to the exponential increase in the size of matrices and of the number of measurement outcomes as the number of qubits increase, the use of JAX [23] has been speeding up the time of reconstruction of rho_shadow. Here we will be using jax.numpy as jnp. In order to reconstruct the density matrix we will be using (6). We define the inverse map as Minv for N qubits and a matrix X and this is given by:

```
def Minv(N,X):
    return ((2**N+1.))*X - np.eye(2**N)
```

Now we need to find the $U^\dagger$ equivalent to our measurement basis we thus define rotGate which attribute the gate of the unitary transformation $U$ from a measurement schedule.

```
def rotGate(g):
    if g=="X":
        return 1/np.sqrt(2)*np.array([[1.,1.],[1.,-1.]])
    elif g=="Y":
        return 1/np.sqrt(2)*np.array([[1.,-1.0j],[1.,1.j]])
    elif g=="Z":
        return np.eye(2)
    else:
        raise NotImplementedError(f"Unknown gate {g}")
```

A last parameter to take into consideration is that the measurement outcomes in Cirq is given in the decimal basis and not the binary basis. We then create the function int_to_binary which translate a number from decimal basis to binary basis. This is simply given by the following code:

```
def int_to_binary(N,b):
    bit = format(b,'b')
    if len(bit)<N:
        for _ in range(N-len(bit)):bit = str(0)+bit
    return bit
```

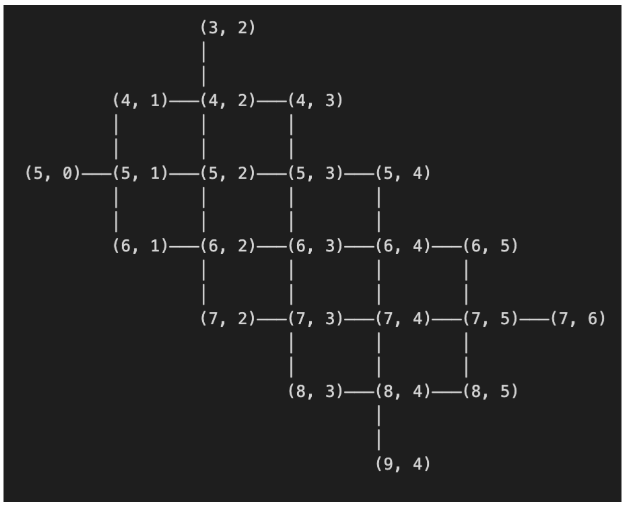
Now that everything have been taking into consideration, we can assembly every part and go through every measurement outcomes from results in order to retrieve the density matrix. This is performed by:

```
def reconstruct(labels,results):
    shadows = jnp.zeros((2**N,2**N),dtype=jnp.complex64)
    shots = 0
    for pauli_string,counts in zip(labels,results):
        for bit,count in counts.items():
            bit = int_to_binary(N,bit)
            mat = 1.
            for i,bi in enumerate(bit):
                b = rotGate(pauli_string[i])[int(bi),:]
                mat = jnp.kron(mat,Minv(1,np.outer(b.conj(),b)))
            shadows+=mat*count
            shots+=count
    return shadows/shots
```

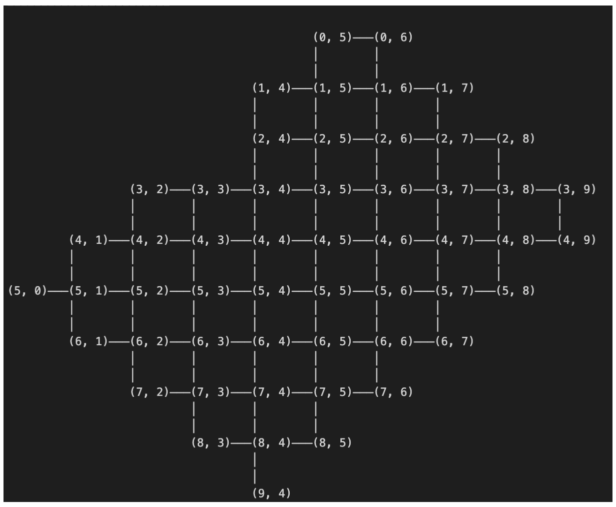
One should note that `b=rotGate(pauli_string[i])[int(bi),:]` correspond to $U_i^s \left|b_i^s\right\rangle$ from (6).

## 4.3 Quantum Virtual Machine

In order to get more realistic noise, we will be using the Quantum Virtual Machine [24] from Google AI. We will be using the two different processor proposed: weber and rainbow. The configuration of the processors are given in Fig.14



(a)



(b)

Figure 14: Representation of the disposition of the qubits of the processor (a) rainbow and (b) weber.

Due to the configuration of the qubits, it is necessary to respect the locality between the qubits, meaning that if two qubits are not connected then no operation involving both of those qubits is possible.

Another thing to take into account is the possible operation admitted by the processor. To resolve both of these problems, we need first to translate the circuit to a circuit using the correct gates. This can be done using a simple function from Cirq given by:

```
translated_ghz_circuit = cirq.optimize_for_target_gateset(
                              c_m, context=cirq.TransformerContext(deep=True),
                              gateset=cirq.SqrtIswapTargetGateset()
                              )
```

where `c_m` is our circuit. Now that we have the correct circuit we need to respect the disposition of our grid of qubits. For that we have to perform the qubit selection:

```
qubits = cirq.LineQubit.range(5)

device_qubit_chain=[
cirq.GridQubit(4, 1),
cirq.GridQubit(4, 2),
cirq.GridQubit(4, 3),
cirq.GridQubit(4, 4),
cirq.GridQubit(4, 5),
]
qubit_map = dict(zip(qubits, device_qubit_chain))
device_ready_ghz_circuit = translated_ghz_circuit.transform_qubits(
                            lambda q: qubit_map[q])
```

where `qubits` are our qubits from the linear five qubits circuit and `device_qubit_chain` are the qubits chosen by hand from the quantum processor. One can note that this correspond to the weber processor since it is impossible to have this configuration with rainbow (see Fig.14). Finally, the last thing to be taken into consideration is the non-uniform distribution of error in the processors. Indeed, some qubits are more impacted by errors than others. The main errors to consider are: the readout error and the entangling errors. The first one is created when the measurement outcomes is giving another result than the one expected and the second is created while performing two qubits gate operation. The error disposition for the readout errors for the rainbow processor is presented in Fig.15 and for the weber processor in Fig.16

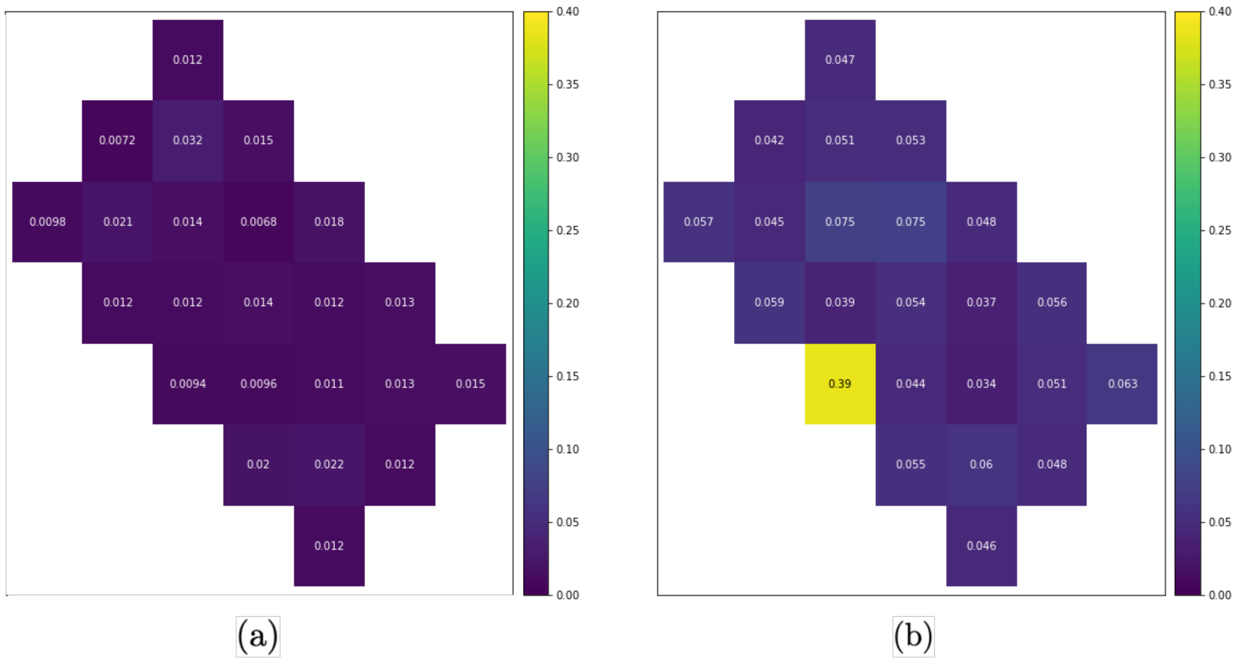
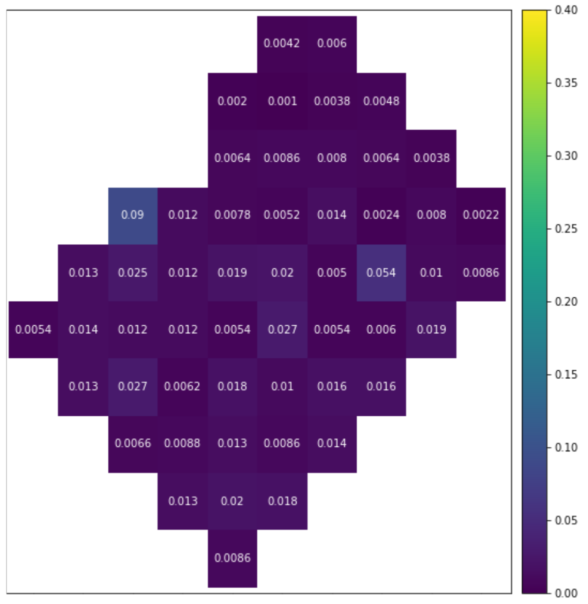
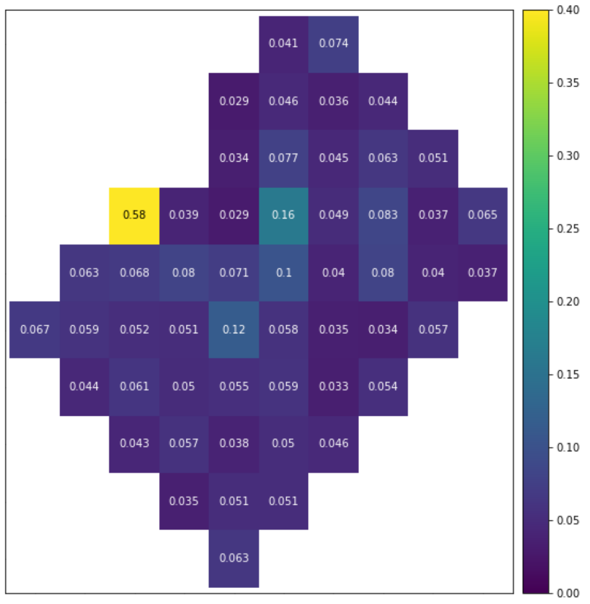


Figure 15: Readout errors for the rainbow processor for (a) $|0\rangle$ measured as $|1\rangle$ and (b) $|1\rangle$ measured as $|0\rangle$.



(a)



(b)

Figure 16: Readout errors for the rainbow processor for (a) $|0\rangle$ measured as $|1\rangle$ and (b) $|1\rangle$ measured as $|0\rangle$.

We observe that for both processor the readout error of $|1\rangle$ measured as $|0\rangle$ is significantly higher. It also permit ourself avoir the highest errors which are 39% of error for the rainbow processor and 58% for the weber processor. Those errors makes the quantum computer not enough advanced to be considered as fault-tolerant and then we can consider this processor as being a NISQ computer.

Now, the entangling error disposition for the rainbow processor is given in Fig.17 and for the weber processor in Fig.18. We will be studying the entangling errors from two different gates: the ISWAP gate and the Sycamore gate. The ISWAP gate between two qubits perform the unitary transformation given by

$$\text{ISWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The Sycamore gate is given by the following unitary transformation

$$\text{Sycamore} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 0 \\ 0 & 0 & 0 & e^{-i\pi/6} \end{pmatrix}.$$

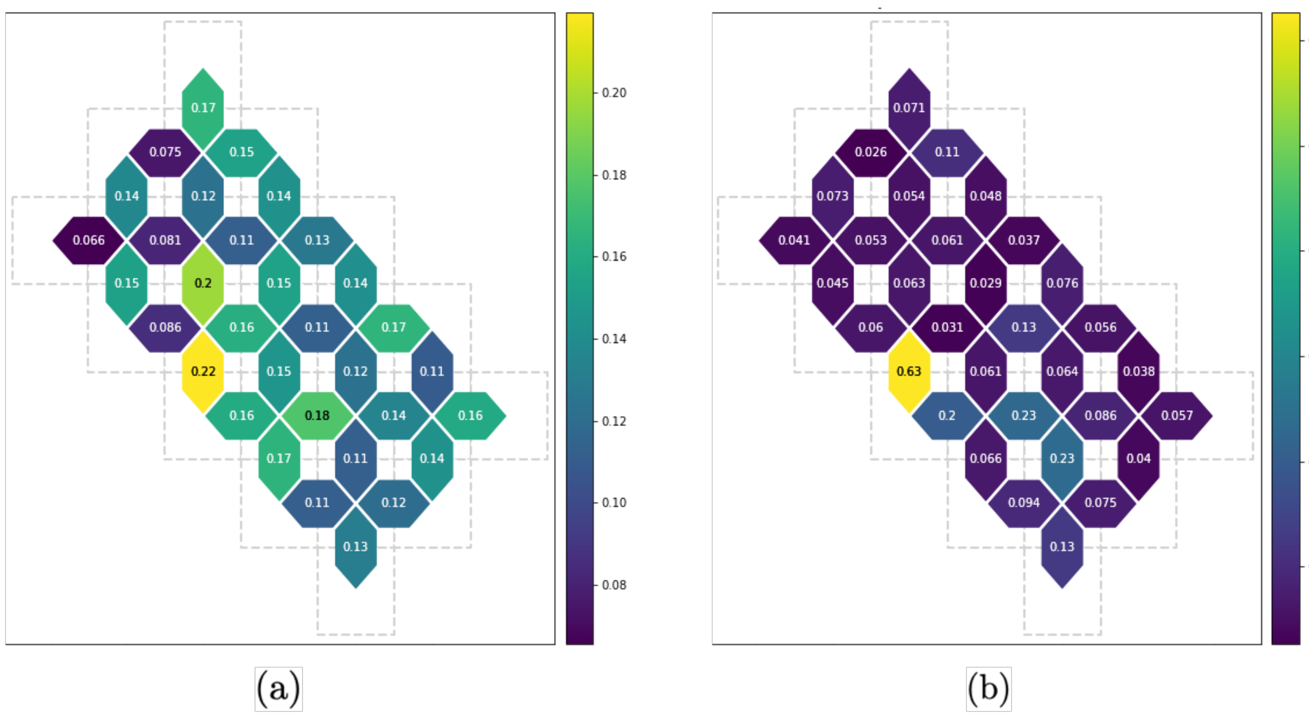


Figure 17: Entangling errors for the rainbow processor for (a) any power of ISWAP gate(b) the Sycamore gate.

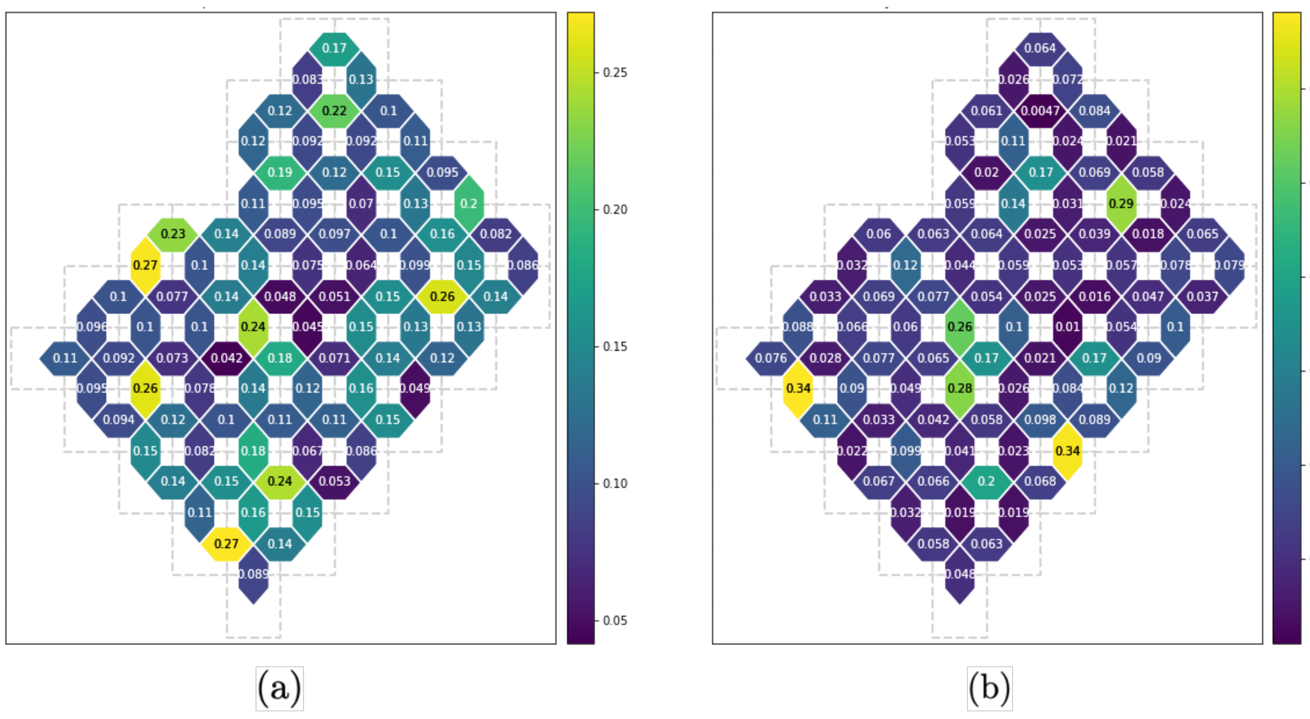


Figure 18: Readout errors for the rainbow processor for (a) any power of ISWAP gate(b) the Sycamore gate.

One can note that the errors using the Sycamore gates in Fig.17 and Fig.18 are relatively lower since the processor is prepared using Sycamore gates

### 4.4 IBM quantum computers

For more realistic noise, we decide to simulate shadow tomography on IBM quantum computers [25]. To reduce the queuing time, we choose the least demanded computer of the 5-qubits quantum computers (such as ibmq-lima). However, we have to take into consideration the limitations set by IBM for the use of their computers. For one job, a computer can perform 100 different circuit with 200000 shots per circuit. The number of shots was not a limitation for our experiment however, the number of circuit per job was. Then, the method used was to split our set of circuit a create a job with each splitted set of circuit. The second limitation, was the use of qiskit for the input circuit. To resolve this problem we used a translator from Cirq to qiskit circuit found in a Mitiq [26] a package for error-mitigation in quantum circuit. In order to translate from Cirq to qiskit we simply have to apply `qc = to_qiskit(c_c)` where `qc` is the qiskit circuit and `c_c` the Cirq circuit. We present the code to run our set of circuit already translated to qiskit on IBM quantum computers. Here we choosed to run our code on the computer ibmq-lima:

```
def run_batch(circuit_array,count):
    circuit_array = np.split(np.array(circuit_array),len(circuit_array)//100+1)
    results =[]
    for circ in circuit_array:

        IBMQ.save_account("YOUR TOKEN", overwrite=True)
        provider = IBMQ.load_account()

        provider = IBMQ.get_provider(hub='ibm-q', group='open', project='main')
        backend = provider.get_backend('ibmq_lima')
        mapped_circuit = [transpile(ci, backend=backend) for ci in circ]
        qobj = assemble(mapped_circuit, backend=backend, shots=count)

        job = backend.run(qobj)
        result = job.result()
        counts = result.get_counts()
        for c in counts: results.append(c)
    return results
```

We first start by splitting our circuit list. Then, we open our IBMQ account with our personal token. We choose the computer or backend with `provider.get_backend`. Here we chose ibmq-lima. We prepare our circuit by using `transpile` and `assemble`. Finally we run our splitted list of circuit using the backend and retrieve the result in a dictionary. In order to reconstruct the density matrix we use the same `reconstruct` function than the shadow tomography method using Cirq.

### 4.5 Multiprocessing

Due to the long simulation time, we use multiprocessing [27]. For that we will be separating our code in $N_m$ part and run it in parallel. We will be using it to shorten the simulation time of the $[[10, 2, 3]]$ code. We will be using it in order to generate the results and reconstruct the density matrix as seen in Sec.4.2.

## 5 Conclusion

In this report, we introduced two methods for quantum error mitigation - virtual distillation and subspace expansion - and evaluated their performance on real and simulated quantum computers. We implemented both methods via shadow tomography, a sample-efficient procedure for estimating observables and performing state reconstruction. In our first experiment, we performed virtual distillation on a five-qubit GHZ state using the IBMQ Lima quantum computer and Google Quantum Virtual Machine. We found that for the depolarizing noise, virtual distillation could mitigate the errors and retrieve relatively accurately the initial state for low depolarizing noise rate ($p = 0.05$). For the IBMQ Lima quantum computer and Google Quantum Virtual Machine, the results obtained showed that virtual distillation is well adapted to their noise model and permit retrieving the correct states. We noted however, the difficulty to mitigate the errors for observables acting on all qubits such as $\prod_i X_i$. In our second experiment, we performed subspace expansion and virtual distillation on a two-qubit logical Bell state using the Google Quantum Virtual Machine. We encoded the logical state using the [[5, 1, 3]] code and found that virtual distillation is not enough to retrieve the correct observable average values for low power using depolarizing noise model. However, coupled to virtual distillation the correct average value is retrieved for low power. When using the Quantum Virtual Machine from Google AI, virtual distillation alone does not permit retrieving the correct average value for any observable even at infinite power. However, using subspace expansion, it is possible retrieving the correct average value at high power. As previous researches, one could be interested into performing the two logical qubits encoded in the $[[5, 1, 3]]$ error correcting code on the IBM quantum computers. Also, we could project our interest on other error correcting code such as the floquet codes [28] such as the ladder code or honeycomb code [29].

## References

[1] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, M. Head-Gordon, "Simulated Quantum Computation of Molecular Energies", https://www.science.org/doi/10.1126/science.1113479

[2] Harrigan, M.P., Sung, K.J., Neeley, M. et al, "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor", Nat. Phys. 17, 332–336 (2021). https://doi.org/10.1038/s41567-020-01105-y

[3] Peruzzo, A., McClean, J., Shadbolt, P. et al., "A variational eigenvalue solver on a photonic quantum processor", Nat Commun 5, 4213 (2014), https://doi.org/10.1038/ncomms5213

[4] Google AI Quantum and Collaborators*† et al., "Hartree-Fock on a superconducting qubit quantum computer", Science369,1084-1089(2020).https://www.science.org/doi/10.1126/science.abb9811

[5] H.Y. Huang, R. Kueng and J. Preskill, "Predicting Many Properties of a Quantum System from Very Few Measurements" https://www.nature.com/articles/s41567-020-0932-7

[6] H. Hu, R. LaRose, Y. You, E. Rieffel, Z. Wang, "Logical shadow tomography: Efficient estimation of error-mitigated observables" https://arxiv.org/abs/2203.07263

[7] A. Seif, Z. Cian, S. Zhou, S. Chen, L. Jiang,"Shadow Distillation: Quantum Error Mitigation with Classical Shadows for Near-Term Quantum Processors", https://arxiv.org/abs/2203.07309

[8] McClean, J.R., Jiang, Z., Rubin, N.C. et al., "Decoding quantum errors with subspace expansions", Nat Commun 11, 636 (2020). https://doi.org/10.1038/s41467-020-14341-w

[9] N. Yoshioka, H. Hakoshima, Y. Matsuzaki, Y. Tokunaga, Y. Suzuki, S. Endo "Generalized quantum subspace expansion", https://arxiv.org/abs/2107.02611

[10] R. Levy D. Luo and B.K. Clark, "Classical Shadows for Quantum Process Tomography on Near-term Quantum Computers" https://arxiv.org/abs/2110.02965

[11] W. J. Huggins, S. McArdle, T. E. O'Brien, J. Lee, N. C. Rubin, S. Boixo, K. B. Whaley, R. Babbush and J. R. McClean,"Virtual Distillation for Quantum Error Mitigation", https://arxiv.org/abs/2011.07064

[12] N. Yoshioka, H. Hakoshima, Y. Matsuzaki, Y. Tokunaga, Y. Suzuki, and S. Endo,"Generalized quantum subspace expansion", https://arxiv.org/abs/2107.02611

[13] Y. Le Fur, R. LaRose, "Dynamically Generated logical Qubits: Ladder Code"

[14] McClean, J.R., Jiang, Z., Rubin, N.C. et al, "Decoding quantum errors with subspace expansions", Nat Commun 11, 636 (2020). https://doi.org/10.1038/s41467-020-14341-w

[15] Z. Eldredge, M. Foss-Feig, J. A. Gross, S. L. Rolston, and A. V. Gorshkov "Optimal and secure measurement protocols for quantum sensor networks", https://link.aps.org/doi/10.1103/PhysRevA.97.042337

[16] Amrit De and Leonid P.Pryadko, "Universal set of Dynamically Protected Gates for Bipartite Qubit Networks II: Soft Pulse Implementation of the [[5,1,3]] Quantum Error Correcting Code." https://arxiv.org/abs/1509.01239

[17] T. J. Yoder, R. Takagi, I. L. Chuang, "Universal fault-tolerant gates on concatenated stabilizer codes" https://arxiv.org/abs/1603.03948

[18] C. Ryan-Anderson, N. C. Brown, M. S. Allman, B. Arkin, G. Asa-Attuah, C. Baldwin, J. Berg, J. G. Bohnet, S. Braxton, N. Burdick, J. P. Campora, A. Chernoguzov, J. Esposito, B. Evans, D. Francois, J. P. Gaebler, T. M. Gatterman, J. Gerber, K. Gilmore, D. Gresh, A. Hall, A. Hankin, J. Hostetter, D. Lucchetti, K. Mayer, J. Myers, B. Neyenhuis, J. Santiago, J. Sedlacek, T. Skripka, A. Slattery, R. P. Stutz, J. Tait, R. Tobey, G. Vittorini, J. Walker, D. Hayes , "Implementing Fault-tolerant Entangling Gates on the Five-qubit Code and the Color Code", https://arxiv.org/abs/2208.01863

[19] O. Bettermann and C. Tat Ngai, "Density Matrix and State Tomography", chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://qudev.phys.ethz.ch/static/content/QSIT16/talks/densityMatrix_stateTomography_ChunTat_Oscermann.pdf

[20] R. Blume-Kohout, "Optimal, reliable estimation of quantum states", https://arxiv.org/abs/quant-ph/0611080

[21] Cirq, An open source framework for programming quantum computers, https://quantumai.google/cirq

[22] Qsim, Optimized quantum circuit simulators, https://quantumai.google/qsim

[23] Jax Quickstart, https://jax.readthedocs.io/en/latest/notebooks/quickstart.html

[24] Google AI, Quantum Virtual Machine, Emulates a Google quantum computer https://quantumai.google/cirq/simulate/quantum_virtual_machine

[25] IBM, Compute resources, access IBM Quantum systems and simulators https://quantum-computing.ibm.com/

[26] R. LaRose, A. Mari, S. Kaiser, P. J. Karalekas, A. A. Alves, P. Czarnik, M. El Mandouh, M. H. Gordon, Y. Hindy, A. Robertson, P. Thakre, M. Wahl, D. Samuel, R. Mistri, M. Tremblay, N. Gardner, N. T. Stemen, N. Shammah, W. J. Zeng, "Mitiq: A software package for error mitigation on noisy quantum computers", https://arxiv.org/abs/2009.04417

[27] multiprocessing, a package that supports spawning processes using an API similar to the threading module, https://docs.python.org/fr/3.8/library/multiprocessing.html

[28] Error correction Zoo, Floquet codes, https://errorcorrectionzoo.org/c/floquet

[29] M. B. Hastings, J. Haah, "Dynamically Generated Logical Qubits", https://arxiv.org/abs/2107.02194

## A Code to generate the GHZ state for the $[[5,1,3]]$ code

### A.1 encoding

We present the encoding of the five-qubits error correcting code presented in [16]. Here we explicit the code used using in order to encode the $[[5,1,3]]$ code.

```python
def encoding_5qubits(state,qubits):
    CZ =  cirq.ControlledGate(sub_gate=cirq.Z, num_controls=1)
    CY = cirq.ControlledGate(sub_gate=cirq.Y,num_controls=1)
    circuit = cirq.Circuit()
    if state == "1":
        circuit.append(cirq.X(qubits[4]))
    circuit.append(cirq.H(qubits[0]))
    circuit.append(cirq.H(qubits[1]))
    circuit.append(cirq.H(qubits[2]))
    circuit.append(cirq.H(qubits[3]))
    circuit.append(cirq.CNOT(qubits[0],qubits[4]))
    circuit.append(cirq.CNOT(qubits[1],qubits[4]))
    circuit.append(cirq.CNOT(qubits[2],qubits[4]))
    circuit.append(cirq.CNOT(qubits[3],qubits[4]))
    circuit.append(CZ(qubits[0],qubits[1]))
    circuit.append(CZ(qubits[1],qubits[2]))
    circuit.append(CZ(qubits[2],qubits[3]))
    circuit.append(CZ(qubits[3],qubits[4]))
    circuit.append(CZ(qubits[0],qubits[4]))
    return circuit
```

Once we encoded the state in $|0\rangle_L$ or $|1\rangle_L$ we get the following density matrices given in Fig.19

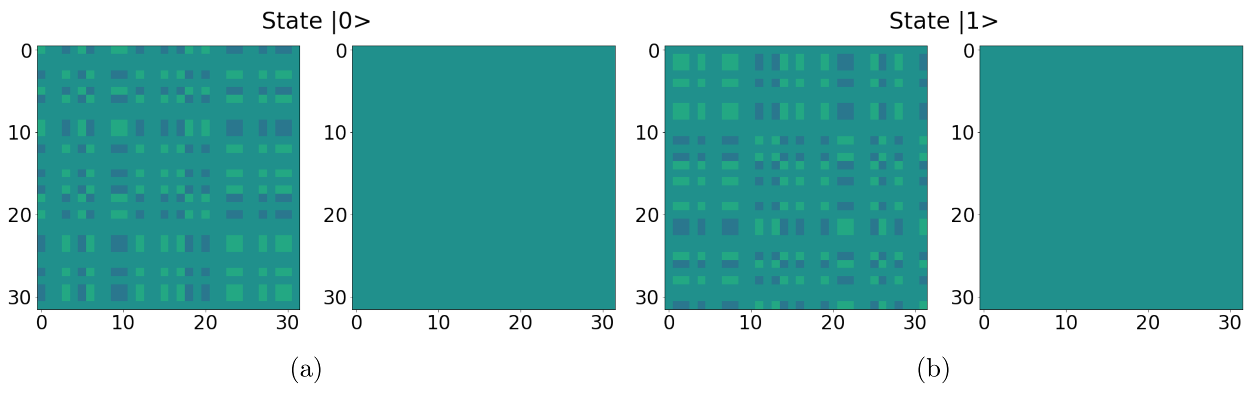


Figure 19: (Density matrix optained with the (a) logical state $|0\rangle_L$ and (b) logical state $|1\rangle_l$ encoded in the $[[5,1,3]]$ error correcting code.

## A.2 Logical Hadamard

The logical hadamard gate for the $[[5,1,3]]$ code presented in [17] is implemented using Cirq and given by :

```
def Hadamard_logical(circuit,qubits):
    for i in range(5):circuit.append(cirq.H(qubits[i]))
    circuit.append(cirq.SWAP(qubits[0],qubits[1]))
    circuit.append(cirq.SWAP(qubits[1],qubits[4]))
    circuit.append(cirq.SWAP(qubits[4],qubits[3]))
    pass
```

In order to test the logical hadamard gate $H_L$, we perform a comparison between the state $(|0\rangle_L + |1\rangle_L)/\sqrt{2}$ and the state $\mathrm{H}_L\,|0\rangle_L$. We present the graphical representation of corresponding density matrix in Fig.20

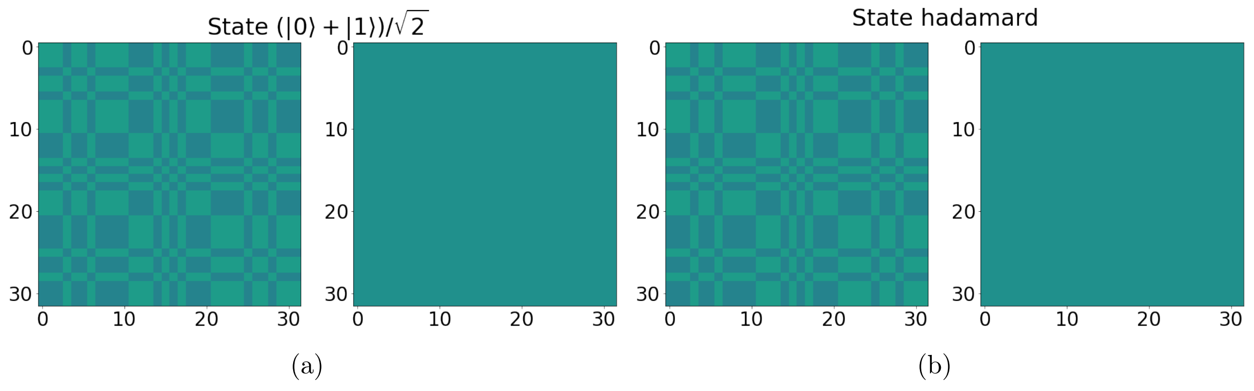


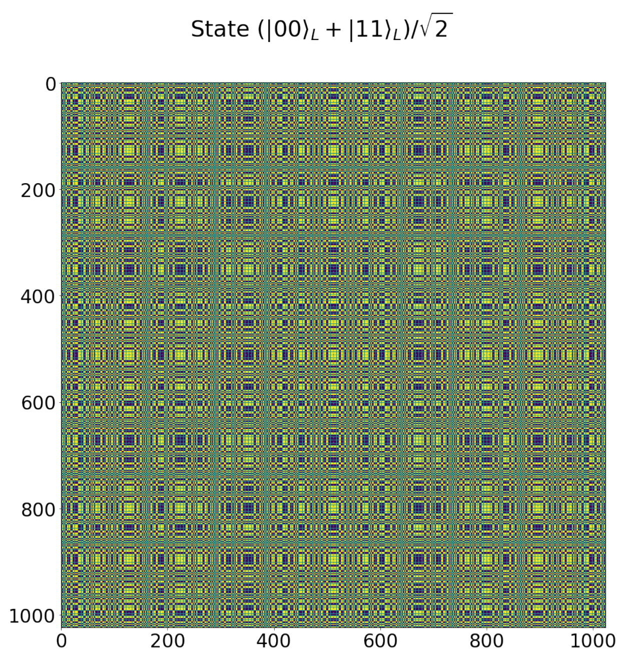
Figure 20: Comparison between (a) the artificially superposed state and (b) the superposed state using logical hadamard gate.

## A.3 Logical CNOT

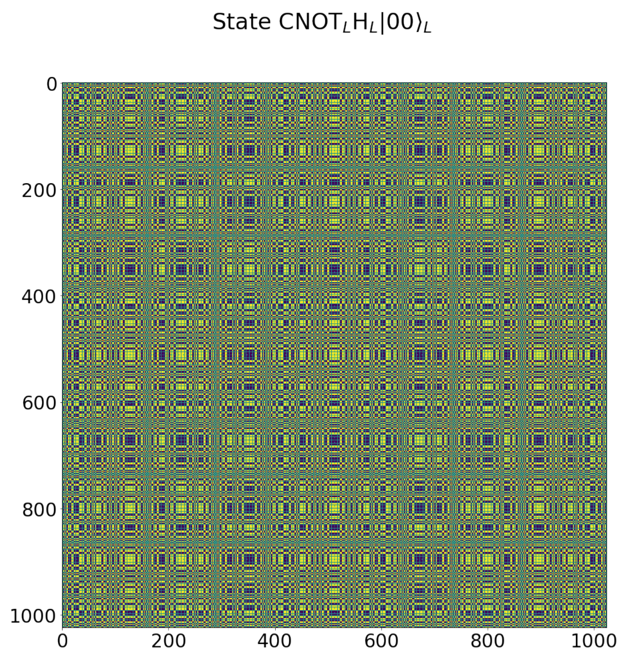
Finally, the last element to construct a GHZ state is the logical CNOT gate. We implement the CNOT gate from [18] using Cirq.

```
def CNOT_logical(circuit,qubits):
    sdg = cirq.S**-1

    circuit.append(cirq.H(qubits[0]))
    circuit.append(cirq.S(qubits[0]))
    circuit.append(cirq.Y(qubits[2]))
    circuit.append(cirq.H(qubits[4]))
    circuit.append(cirq.S(qubits[4]))

    circuit.append(cirq.H(qubits[5]))
    circuit.append(cirq.S(qubits[5]))
    circuit.append(cirq.Y(qubits[7]))
    circuit.append(cirq.H(qubits[9]))
    circuit.append(cirq.S(qubits[9]))

    circuit.append(cirq.CNOT(qubits[0],qubits[5]))
    circuit.append(cirq.CNOT(qubits[2],qubits[7]))
    circuit.append(cirq.CNOT(qubits[4],qubits[9]))
    circuit.append(cirq.CNOT(qubits[0],qubits[9]))
    circuit.append(cirq.CNOT(qubits[2],qubits[5]))
    circuit.append(cirq.CNOT(qubits[4],qubits[7]))
    circuit.append(cirq.CNOT(qubits[0],qubits[7]))
    circuit.append(cirq.CNOT(qubits[2],qubits[9]))
    circuit.append(cirq.CNOT(qubits[4],qubits[5]))

    circuit.append(sdg(qubits[0]))
    circuit.append(cirq.H(qubits[0]))
    circuit.append(cirq.Y(qubits[2]))
    circuit.append(sdg(qubits[4]))
    circuit.append(cirq.H(qubits[4]))

    circuit.append(sdg(qubits[5]))
    circuit.append(cirq.H(qubits[5]))
    circuit.append(cirq.Y(qubits[7]))
    circuit.append(sdg(qubits[9]))
    circuit.append(cirq.H(qubits[9]))
    pass
```

In order to check the logical CNOT gate, we choose to compare the artificially constructed state $(|00\rangle_L + |11\rangle_L)/\sqrt{2}$ using Kronecker product and the [[5, 1, 3]] encoding code and the state $\text{CNOT}_L \text{H}_L |00\rangle_L$. We note that the last state $|00\rangle_L$ has been encoded using two concatenated encoding of the [[5, 1, 3] code. We present the corresponding density matrices in Fig.21



(a)



(b)

Figure 21: Comparison between (a) the artificially superposed GHZ state and (b) the GHZ state using logical hadamard and CNOT gates.