UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Master degree in Physics

Development of an open-source calibration framework for superconducting qubits

Supervisor:
Prof. Dr. Stefano Carrazza

Co-supervisor:
Dr. Alessandro Candido

Co-supervisor:
Dr. Andrea Pasquale

Co-supervisor:
Dott. Edoardo Pedicillo

Elisa Stabilini
Matricola n° 28326A
A.A. 2024/2025

Contents

Summary

Chapter 1

Notes on quantum computing

1.1 Qubits

1.2 Operation on qubits

1.2.1 Density matrix

1.3 Quantum operations

A quantum operation is a mathematical transformation that describes how a quantum state changes as a consequence of a physical process. Formally, it is a map $\mathcal{E}$ that transforms a quantum state described by a density operator $\hat{\rho}$ into another state described by a new density operator $\hat{\rho}'$:

$$\mathcal{E}(\rho) = \rho'. \tag{1.1}$$

The simplest example of a quantum operation is the evolution of a quantum state $\hat{\rho}$ of a closed quantum system, under a unitary operator $\hat{U}$, which can be written as $\mathcal{E} \equiv \hat{U}\hat{\rho}\hat{U}^\dagger$.

Depolarizing chennel A depolarizing channel describes a process in which the current state of the n-qubit system ρ, is replaced by $\frac{\mathbb{I}}{2^n}$, with probability d. This process can be represented with a quantum map as follows:

$$\mathcal{E}_{dc}(\rho) = d\frac{\mathbb{I}}{2^n} + (1-d)\rho \tag{1.2}$$

1.4 Superconducting qubits

Chapter 2

Qibo

Tutti i risultati che sono presentati nel seguito sono stati ottenuti utilizzando il software di `Qibolab`per l'interazione con gli strumenti del laboratorio e `Qibocal`per il controllo delle operazioni sui qubit. L'hardware è un chip di QunatumWare. Durante il lavoro condotto per questo progetto di tesi entrambe le libererie, sia Qibocal che Qibolab undergo update and release, for this reason the first part of this work was realized using `Qibocal`v0.1 and `Qibolab`v0.1 while the second part of the work, dato che puntava anche allo sviluppo di routine ch epotessero essere utili per la calibrazione dei qubit è stato realizzato direttamente con `Qibocal`v0.2 e `Qibolab`v0.2.

Chapter 3

RB fidelity optimization

3.0.1 Randomized Benchmarking

A strong limitation to the realization of quantum computing technologies is the loss of coherence that happens as a consequence of the application of many sequential quantum gates to to the quibts. A possible approach to characterize gate error is the quantum process tomography which allows the experimenter to establish the behaviour of a quantum gates; the main drawback of this approach is that process tomography can be very time consumig since its time complexity scales exponentially with the number of qubits involved [1] and the result is affected by state preparation and measurements (SPAM) errors.

To overcome these limitations, randomized benchmarking (RB) was introduced and is currently widely used to quantify the avarage error rate for a set of quantum gates.

The main idea is that the error obtained from the combined action of random unitary gates drawn from a uniform distribution with respect to the Haar measure [2] and applied in sequence to the qubit will avarage out to behave like a depolarizing channel [3]. This last consideration simplifies the characterization of noise because it removes dependence on specific error structures and allows fidelity to be extracted through a simple exponential decay.

It was later shown that it is possible to simplify this procedure even more, by restricting the unitaries to gates in the Clifford group [1] and by not requiring that the sequence is strictly self-inverting [4].

The fundamental principle of RB is the application of sequences of randomly selected quantum gates from the Clifford group $\mathcal{C}$ followed by an inversion gate which, in absence of noise, return the system to its initial state. For real systems, where noise is present, the observed survival probability provides an estimate of the avarage gate fidelity.

The standard RB protocols consist of the following steps:

1. Initialize the system in ground state $|0\rangle$
2. For each sequence-length m build a sequence of m randomly drawn Clifford gates $C_1, C_2, ..., C_m$
3. Determine the inverse gate $C_{m+1} = (C_m \circ ... \circ C_1)^{-1}$
4. Measure $C_{m+1} \circ C_m \circ ... \circ C_1 |0\rangle$

The process must be repeated for multiple sequence of the same length and with varying length.

In ideal systems without noise we should have

$$C_{m+1} \circ C_m \circ ... \circ C_1 |0\rangle = (C_m \circ ... \circ C_1)^{-1} \circ (C_m \circ ... \circ C_1) |0\rangle = |0\rangle \tag{3.1}$$

In realsystems, where noise is present, eq. **??** does not hold; instead randomization with Clifford gates behave as a depolarizing channel 1.2 with depolarization probability d. The survival probability of the initial state $|0\rangle$ for different sequence lengths follows the exponential decay model

$$F(m) = Ap^m + B, \tag{3.2}$$

where $1 - p$ is the rate of depolarization and A and B capture the state preparation and measurement error but not the rate of decay p. Note that the exponential form arises naturally due to the assumption that each gate introduces independent noise.

[1]unitary rotations mapping the group of Puali operators in itself

The parameter p is directly related to the depolarization probability d through the avarage gate fidelity F which, for a depolarizing channel, is given by

$$F = 1 - \frac{d}{2^n - 1}. \tag{3.3}$$

For the details of the calculations to obtain eq. 3.3 see Appendix A (5).

Now we can derive the avarage error per Clifford gate $\epsilon_{Clifford}$

$$\epsilon_{Clifford} = 1 - F, \tag{3.4}$$

where F is the avarage gate fidelity. Sobstituting in 3.4 the formula for the avarage gate fidelity 3.3 we obtain

$$\epsilon_{Clifford} = \frac{d}{2^n - 1} = \frac{1-p}{1-2^{-n}}, \tag{3.5}$$

which shows how the avarage error per Clifford gate is directly connected to the exponential decay rate.

QUA Randomized Benchmarking

For the results we present in the following the technique used slightly differs from the one described in section 3.0.1

3.0.2 Optimization methods

I primi metodi che abbiamo provato per l'ottimizzazione dei parametri sono quelli standard implementati nella libreria `Scipy` [5] evitando metodi gradient-based considerato il landscape potenzialmente complicato della funzione RB. The first gradient-free optimization method to be tested was Nelder-Mead since in letteratura era già stato riportato il suo utilizzo per obiettivi simili [6].

The Nelder-Mead optimization method, originally introduced by Nelder and Mead in 1965 [7], is a widely used numerical optimization technique for unconstrained problems in multidimensional spaces. This derivative-free method is operates using simplex, which is a polytope of $n+1$ vertices in a n-dimensional space. The algorithm iteratively updates the simplex by replacing its worst-performing vertex with a new candidate point, thereby guiding the search towards an optimal solution. If the goal is to minimize a given function $f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^n$ the algorithms proceeds with the following steps:

1. If not otherwise initialized, $n+1$ points are sampled for building the initial symplex
2. `Order` the test points according to their values at vertices: $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_{n+1})$ and check whether the algorithm should terminate.
3. `Calculate` $\mathbf{x}_0$, the centroid of all points except $\mathbf{x}_{n+1}$.
4. `Reflection`: Compute the reflected point $\mathbf{x}_r = \mathbf{x}_0 + \alpha(\mathbf{x}_0 - \mathbf{x}_{n+1})$ with $\alpha > 0$. If $\mathbf{x}_r$ satisfies $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$, then a new simplex is obtained by replacing the worst-performing point $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$ and then go to step 1.
5. `Expansion`: If $\mathbf{x_r}$ is the current best point, meaning that $f(\mathbf{x}_r) < f(\mathbf{x}_1)$, then the expanded point is computed: $\mathbf{x}_e = \mathbf{x}_0 + \gamma(\mathbf{x}_r - \mathbf{x}_0)$ with $\gamma > 1$. If $\mathbf{x}_e$ satisfies $f(\mathbf{x}_e) < f(\mathbf{x}_r)$, then a new simplex is obtained by replacing $\mathbf{x}_{n+1}$ with the expanded point $\mathbf{x}_e$ and then go to step 1.
   If instead $f(\mathbf{x}_e) \geq f(\mathbf{x}_r)$, the new simplex is obtained by replacing $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$, and then go to step 1.
6. `Contraction`: In this case is certain that $f(\mathbf{x}_r) \geq f(\mathbf{x}_n)$ then:
   - If $f(\mathbf{x}_r) < f(\mathbf{x}_{n+1})$: compute the contracted point $\mathbf{x}_c = \mathbf{x}_0 + \rho(\mathbf{x}_r - \mathbf{x}_0)$ with $0 < \rho \leq 0.5$. If $\mathbf{x}_c$ satisfies $f(\mathbf{x}_c) < f(\mathbf{x}_r)$, then a new simplex is obtained by replacing $\mathbf{x}_{n+1}$ with $\mathbf{x}_c$ and go to step 1.
     Else go to step 6.
   - If $f(\mathbf{x}_r) \geq f(\mathbf{x}_{n+1})$: compute the contracted point $\mathbf{x}_c = \mathbf{x}_0 + \rho(\mathbf{x}_{n+1} - \mathbf{x}_0)$ with $0 < \rho \leq 0.5$. If $\mathbf{x}_c$ satisfies $f(\mathbf{x}_c) < f(\mathbf{x}_{n+1})$, the a new simplex is constructed with $\mathbf{x}_c$ and go to step 1.
     Else go to step 6.
7. `Shrinkage`: Replace all points except the best, $\mathbf{x}_1$, with $\mathbf{x}_i = \sigma(\mathbf{x}_i - \mathbf{x}_1), 0 < \sigma \leq 0.5$

The algorithm terminates when the standard deviation of the function values of the current simplex fall below a user-initialized tolerance. When the cycle stops the point of the simplex associated to the lower function value is returned as proposed optimum

The values of the parameters α, γ, ρ and σ were left to default of `scipy`: $\alpha = 1, \gamma = 2, \rho = 0.5, \sigma = 0.5$.

`CMA-ES` Covariance Matrix Adaptation Evolution Strategy (`CMA-ES`) [8], is a population-based evolutionary algorithm designed for optimizing complex, non-convex, and high-dimensional functions.
It belongs to the broader class of Evolution Strategies (ES), a subset of Evolutionary Algorithms (EAs)(see [9]), and is particularly effective for black-box optimization where gradient information is unavailable.

Evolution Strategies (ES) are a class of optimization methods that employ self-adaptive mechanisms to explore the search space efficiently. Unlike classical optimization techniques that rely on gradient descent, ES leverage stochastic sampling to navigate rugged and multimodal landscapes. In this context, CMA-ES is an adaptive stochastic search method that iteratively refines a probability distribution over the search space. Unlike traditional Genetic Algorithms (GAs), which rely on crossover and mutation operators, CMA-ES employs a multivariate normal distribution to generate candidate solutions. The method adaptively updates the distribution's mean and covariance matrix based on the fitness of sampled points.

The fundamental idea behind CMA-ES is the use of a multivariate Gaussian distribution to model promising search directions. Let μ_t denote the mean of the distribution at iteration t, and Σ_t the covariance matrix. Then, a new population of λ candidate solutions $\mathbf{x}_i^(t+1) \sim \mu_t + \sigma_t \mathcal{N}(0, \Sigma_t)$, where σ_t is a step size controlling the exploration.

The CMA-ES algorithm follows the following steps:

1. If not otherwise specified, the initial parameters are set: mean vector $\mu_{\mathbf{0}}$, covariance matrix Σ_0[2], step size σ_0, population size λ

2. Generate λ new candidate solutions $\mathbf{x}_i$ according to a multivariate normal distribution.

3. Evaluate the objective function $f(\mathbf{x}_i)$ for each candidate solution.

4. Sort the new candidate solutions based on fitness: $f(\mathbf{x}_0) \leq ... \leq f(\mathbf{x}_\lambda)$.

5. Update the mean vector μ with the $m = \lfloor \lambda/2 \rfloor$ top performing solutions:

$$\mu \leftarrow \sum_{i=0}^{m} \mathbf{w}_i \mathbf{x}_1, \tag{3.6}$$

   where $\mathbf{w}_i$ are internally defined weights.

6. Update the isotropic and anisotropic evolution path $\mathbf{p}_\sigma$, $\mathbf{p}_c$ [3].

7. Update the covariance matrix:

$$C \leftarrow (1 - c_1 - c_\mu)C + c_1 \mathbf{p}_c \mathbf{p}_c^T + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_i \mathbf{y}_i^T, \tag{3.7}$$

   where c_1 and c_μ are learning rates and $\mathbf{y}_i$ represents the deviation of the i-th cnadidate solution from the mean **mu**.

8. Update the step size using a cumulative path evolution mechanism

$$\sigma \leftarrow \sigma \cdot \exp\left(\frac{c_\sigma}{d_\sigma} \left(\|\mathbf{p}_\sigma\| - E\|\mathcal{N}(0, I)\|\right)\right), \tag{3.8}$$

   where c_σ is the learning rate for step-size adaptation, d_σ is a damping factor $\|\mathbf{p}_\sigma\|$ is the length of the evolution path and $E\|\mathcal{N}(0, I)\|$ is the expected length of a standard normally distributed random vector.

Nel seguito, a meno che non sia diversamente specificato, i parametri sono stati inizializzati ai valori di default della libereria `CMA-ES`

[2] $\Sigma_0 = \mathbb{I}$ for isotropic search

[3] For details on the update process of the evolution paths see [8].

`Optuna` Oltre ai metodi elencati sopra, per l'ottimizzazione si è provato ad utilizzare metodi del tipo TPE, utilizzandone l'implementazione che è stata fatta nella libreria `optuna` [10].

Tree-Structured Parzen Estimator (TPE) is a Sequential Model-Based Optimization (SMBO) approach [11]. SMBO methods sequentially construct models to approximate the performance of optimization parameters based on historical measurements, and then subsequently choose new parameters values to test based on this model. [12] At the heart of SMBO is the idea of building a surrogate model, which is used to predict the objective function's values for unseen parameters configurations. The surrogate model is iteratively updated as new observations are made, and the optimization process balances exploration, which focuses on uncertain regions of the search space, and exploitation, which focuses on areas that are more likely to improve the objective based on past evaluations. This balance ensures that the optimization process makes efficient use of resources and avoids wasting time on suboptimal regions.

The TPE algorithm is a probabilistic model-based optimization method that uses non-parametric density estimation to guide the search. Unlike traditional Bayesian optimization approaches that model the objective function directly, TPE models the probability densities of good and bad parameter configurations independently.

Chapter 4

Gates fine-tunig

4.1 RX90 calibration

Another possible source of error is ...

4.2 Flux pulse correction

4.2.1 Notes on signal analysis

4.2.2 Cryoscope

The experiment that we describe in this section was first introduced in [13], the goal is to determine predistortions that needs to be applied to a flux pulse signal so that the qubit receives the flux pulse as intended by the experimenter.

4.2.3 Filter determination

IIR

FIR

for description and notes on `CMA-ES` see section 3.0.2

Output filters in QM

Chapter 5

Conclusions

Appendix A

Starting with the definition of average gate fidelity:

$$F = \int d\psi \left\langle\psi\right| U^\dagger \mathcal{E}(\left|\psi\right\rangle \left\langle\psi\right|) U \left|\psi\right\rangle$$

The quantum maps that represent a depolarizing channel is

$$\mathcal{E}(\rho) = (1-d)U\rho U^\dagger + d\frac{I}{2^n} \tag{5.1}$$

When we substitute it in the avarage gate fidelity definition we get:

$$\begin{aligned}
F &= \int d\psi \left\langle\psi\right| U^\dagger \left[(1-d)U\left|\psi\right\rangle\left\langle\psi\right| U^\dagger + d\frac{\mathbb{I}}{2^n}\right] U\left|\psi\right\rangle \\
&= \int d\psi \left\langle\psi\right| U^\dagger \left[(1-d)U\left|\psi\right\rangle\left\langle\psi\right| U^\dagger\right] U\left|\psi\right\rangle + \int d\psi \left\langle\psi\right| U^\dagger \left[d\frac{I}{2^n}\right] U\left|\psi\right\rangle \\
&= (1-d)\int d\psi \left\langle\psi|\psi\right\rangle\left\langle\psi|\psi\right\rangle + \frac{d}{2^n}\int d\psi \left\langle\psi\right| U^\dagger \mathbb{I} U\left|\psi\right\rangle \\
&= (1-d) + \frac{d}{2^n}\int d\psi \left\langle\psi|\psi\right\rangle \\
&= (1-d) + \frac{d}{2^n} = 1 - d + \frac{d}{2^n} = 1 - d\left(1 - \frac{1}{2^n}\right) \\
&= 1 - d\frac{2^n - 1}{2^n}
\end{aligned}$$

Bibliography

[1]M. Mohseni, A. T. Rezakhani, and D. A. Lidar, “Quantum-process tomography: resource analysis of different strategies”, Phys. Rev. A **77**, 032322 (2008).

[2]A. A. Mele, “Introduction to haar measure tools in quantum information”, Quantum **8**, 1340 (2024).

[3]J. Emerson, R. Alicki, and K. Życzkowski, “Scalable noise estimation with random unitary operators”, Journal of Optics B: Quantum and Semiclassical Optics **7**, S347 (2005).

[4]E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, “Randomized Benchmarking of Quantum Gates”, en, Physical Review A **77**, arXiv:0707.0963 [quant-ph], 012307 (2008).

[5]P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”, Nature Methods **17**, 261–272 (2020).

[6]J. Kelly, R. Barends, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, I.-C. Hoi, E. Jeffrey, A. Megrant, J. Mutus, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, A. N. Cleland, and J. M. Martinis, “Optimal Quantum Control Using Randomized Benchmarking”, en, Physical Review Letters **112**, 240504 (2014).

[7]J. A. Nelder and R. Mead, “A simplex method for function minimization”, The Computer Journal **7**, 308–313 (1965).

[8]M. Nomura and M. Shibata, *Cmaes : a simple yet practical python library for cma-es*, 2024.

[9]A. N. Sloss and S. Gustafson, *2019 evolutionary algorithms review*, 2019.

[10]T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: a next-generation hyperparameter optimization framework”, in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining (2019).

[11]F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration”, in Learning and intelligent optimization, edited by C. A. C. Coello (2011), pp. 507–523.

[12]B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, “Taking the human out of the loop: a review of bayesian optimization”, Proceedings of the IEEE **104**, 148–175 (2016).

[13]M. A. Rol, L. Ciorciaro, F. K. Malinowski, B. M. Tarasinski, R. E. Sagastizabal, C. C. Bultink, Y. Salathe, N. Haandbaek, J. Sedivy, and L. DiCarlo, “Time-domain characterization and correction of on-chip distortion of control pulses in a quantum processor”, en, Applied Physics Letters **116**, arXiv:1907.04818 [quant-ph], 054001 (2020).

Acknowledgement

Appendix A

Starting with the definition of average gate fidelity:

$$F = \int d\psi \, \langle\psi| \, U^\dagger \mathcal{E}(|\psi\rangle \langle\psi|) U \, |\psi\rangle$$

The quantum maps that represent a depolarizing channel is

$$\mathcal{E}(\rho) = (1-d)U\rho U^\dagger + d\frac{I}{2^n} \tag{5.2}$$

When we substitute it in the avarage gate fidelity definition we get:

$$\begin{aligned}
F &= \int d\psi \, \langle\psi| \, U^\dagger \left[(1-d)U \, |\psi\rangle \langle\psi| \, U^\dagger + d\frac{\mathbb{I}}{2^n}\right] U \, |\psi\rangle \\
&= \int d\psi \, \langle\psi| \, U^\dagger \left[(1-d)U \, |\psi\rangle \langle\psi| \, U^\dagger\right] U \, |\psi\rangle + \int d\psi \, \langle\psi| \, U^\dagger \left[d\frac{I}{2^n}\right] U \, |\psi\rangle \\
&= (1-d) \int d\psi \, \langle\psi|\psi\rangle \langle\psi|\psi\rangle + \frac{d}{2^n} \int d\psi \, \langle\psi| \, U^\dagger \mathbb{I} U \, |\psi\rangle \\
&= (1-d) + \frac{d}{2^n} \int d\psi \, \langle\psi|\psi\rangle \\
&= (1-d) + \frac{d}{2^n} = 1 - d + \frac{d}{2^n} = 1 - d\left(1 - \frac{1}{2^n}\right) \\
&= 1 - d\frac{2^n - 1}{2^n}
\end{aligned}$$