UNIVERSITÀ DEGLI STUDI DI MILANO

Corso di Dottorato in Fisica, Astrofisica e Fisica Applicata
Ciclo XXXVIII
Dipartimento di Fisica

A Full-Stack Software Architecture for the Control and Calibration of Quantum Hardware

Settore Scientifico Disciplinare FIS/02

Supervisore: Stefano Carrazza

Coordinatore: Aniello Mennella

Tesi di Dottorato di:
Edoardo Pedicillo

Matricola:
R13831

ORCID:
0009-0007-7974-9469

Anno Accademico 2024-2025

Commission of the final examination:

External Referee: Caterina Braggio
External Referee: Juan Manuel Cruz Martinez

External Member: German Sierra
External Member: Fabio Maltoni
External Member: Angelo Nucciotti

Final examination:

Date 28 November 2025

Università degli Studi di Milano, Dipartimento di Fisica, Milano, Italy

Cover illustration:

Image by Silvia Pedicillo.

MIUR subjects:

FIS/02 - Fisica Teorica, Modelli e Metodi Matematici

PACS:

03.67.Lx, 07.05.Hd, 07.05.Bx, 89.20.Ff

Keywords:

Quatum computing, Quantum simulation, Quantum control, Quantum calibration

To Gaja

Abstract

Over the last 20 years, thanks to the development of quantum technologies, it has been possible to deploy quantum algorithms and applications that before were only accessible through simulation on real quantum hardware. The current devices available are often referred to as noisy intermediate-scale quantum (NISQ) computers, and they require calibration routines in order to obtain consistent results.

In this context, we present `Qibo`, an open-source framework for quantum computing. `Qibo` was initially born as a tool for simulating quantum circuits. Through its modular layout for backend abstraction, it is possible to change effortlessly between different backends, including a simulator based on just-in-time compilation, `Qibojit`.

In order to enable the execution and calibration of self-hosted quantum hardware we have developed two open-source libraries integrated with the `Qibo` framework: `Qibolab` and `Qibocal`. `Qibolab` provides the software layer required to automatically execute circuit-based algorithms on custom self-hosted quantum hardware platforms. It enables experimentalists and developers to delegate all complex aspects of hardware implementation to the library so they can standardize the deployment of quantum computing algorithms in a hardware-agnostic way.

`Qibocal` is based on a modular QPU (Quantum Processor Unit) platform agnostic approach and introduces tools that support the calibration and characterization of QPUs on three different levels: development, deployment and distribution. `Qibocal` provides a code library to rapidly develop protocols for different hardware abstraction layers. The integration with `Qibo` allows one to easily switch between hardware execution and high-perfomance simulation.

Finally, to showcase the capabilities of the framework, we present the calibration of a superconducting qubit pair with `Qibocal` and applications developed with `Qibo`, including a novel method for improving ground state preparation through Variational Quantum Eigensolvers and Double Bracket algorithms.

Introduction

Quantum computing is focused on using the principles of quantum mechanics, such as superposition and entanglement, to perform computations that are infeasible or require excessive time in classical devices.

Among the different physical platforms investigated for achieving quantum bits (qubits), superconducting circuits have become one of the most sophisticated and promising technologies. Superconducting qubits are created from electrical circuits made of superconducting materials, usually aluminum or niobium, operating close to absolute zero temperature to remove resistive losses. Throughout all circuits, the transmon is one of the most extensively studied, employing the Josephson junction as a nonlinear component that introduces anharmonicity to the system, a crucial attribute for the efficient control of quantum states.

These devices use circuits at the millimeter scale built with lithographic techniques borrowed from conventional integrated circuits, enabling substantial adaptability in design. An additional advantage is that they function in the microwave regime, a frequency spectrum in which a significant portion of modern technology is already well-established. Their rapid gate times and capacity to couple superconducting circuits via tunable interactions provide the potential of deploying this technology in scalable quantum processors.

Conversely, superconducting qubits need to function at millikelvin temperatures within dilution refrigerators, which are expensive and complex systems. Additionally, the restricted quantity of microwave lines that can be contained into such cryogenic systems could become a limiting factor for scaling the number of qubits.

Quantum computing demands more than just high quality hardware and low-noise transmission lines; it further relies on strong software to effectively manage the electronics and optimize calibration and measurement procedures. This thesis presents the Qibocal and Qibolab libraries, created inside the Qibo ecosystem to meet these requirements. These libraries offer a integrated system, allowing users to effortlessly simulate quantum circuits and execute them on quantum platforms.

This thesis is structured in three parts. The part I introduces the fundamental concepts necessary to understand the manipulation and readout of the quantum state stored in a superconducting chip. The part II defines the concept of a full-stack quantum computer and provides a detailed description of the software developed. The final part III is dedicated to applications. Specifically, the first chapter will discuss the calibration of a quantum platform using Qibocal, including examples taken from an actual chip calibration performed at the Technology Innovation Insti-

tute (TII) in Abu Dhabi. The last chapter will present work on ground state preparation using the double bracket quantum algorithm, where we explored this problem with an algorithm designed to be executable on current quantum chips.

Contents

Part I.

Theoretical background

1. Overview

This first part provides a foundational overview of superconducting qubit platforms. It will cover the principles of the Josephson junction as the basis for creating an artificial atom (chapter 2), the advantages of the transmon regime (chapter 3), and techniques for frequency control. This part concludes with an exposition of the methods used for qubit state manipulation (chapter 4), measurement, and the critical topic of environmental coupling (chapter 5).

2. Josephson junction

The fundamental building block of superconducting qubit chips is the Josephson junction (see Fig. 2.1), a nonlinear electrical component consisting of two superconducting pads separated by a thin insulating barrier. At temperatures below the critical value, all Cooper pairs in the superconducting islands can be described by a single macroscopic wavefunction, $\psi = \sqrt{\rho}e^{i\theta}$, where ρ denotes the density of the pairs and θ is the phase common to all particles. The current I and voltage V across the junction are governed by the relations,

$$I = I_C \sin(\phi(t)), \tag{2.1}$$

$$\frac{d\phi(t)}{dt} = \frac{2\pi}{\Phi_0} V(t), \tag{2.2}$$

commonly referred to as the first and second Josephson equations, respectively. Here, $\phi = \theta_1 - \theta_2$ is the phase difference between the two islands, $\Phi_0 = h/2e \approx 2.07 \times 10^{-15}\,\mathrm{Tm}^2$ is the superconducting flux quantum, and I_C is the critical current of the junction, which depends on the system geometry.

Two fundamental phenomena arise from the Josephson relations. If no voltage is applied across the junction terminals ($V(t) = 0$), Eq. 2.2 implies that $\phi(t) = \text{const.}$, and therefore a current can flow without any applied voltage. This is known as the **DC Josephson effect**. Conversely, if a constant voltage V is applied, the phase evolves linearly in time and the current oscillates as

$$I = I_C \sin\left(\phi_0 + \frac{2\pi}{\Phi_0} Vt\right), \tag{2.3}$$

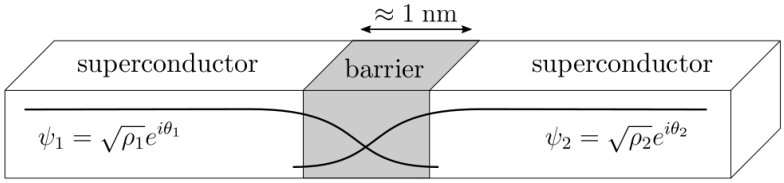


Figure 2.1.: Schematic representation of a Josephson junction. The two superconducting islands are separated by the insulator barrier. The Cooper pairs in the two islands can be described by macroscopic wavefunctions. The image has been adapted from [1].

which is referred to as the **AC Josephson effect**.

From these relations we can also evaluate the effective Josephson inductance L_J,

$$L_J = \frac{V}{dI/dt} = \frac{\Phi_0}{2\pi I_C \cos\phi} = \frac{\Phi_0}{2\pi\sqrt{I_C^2 - I^2}} = \frac{L_{J0}}{\sqrt{I_C^2 - I^2}}, \tag{2.4}$$

with $L_{J0} = \Phi_0/(2\pi I_C)$. We observe that the inductance diverges as $I \to I_C$ and depends nonlinearly on the junction phase. This intrinsic nonlinearity makes the Josephson junction a crucial element for qubit realization.

For completeness, the electrical energy stored in a Josephson junction can be expressed as

$$U = \int_0^t IV\,dt = \int_0^\phi \Phi_0 I_C \sin(\phi')\,d\phi' = -E_J \cos\phi + \text{const}, \tag{2.5}$$

where we have introduced the Josephson energy,

$$E_J = \frac{\Phi_0 I_C}{2\pi}, \tag{2.6}$$

as we will see in the next sections, this parameter is crucial for the properties of the qubits such as the qubit frequency and the coherence times.

If the junction is biased with a current $I > I_C$, superconductivity is destroyed and the junction enters its normal state, exhibiting a resistive behavior. In this regime, Ohm's law applies,

$$V = R_n I, \tag{2.7}$$

where R_n is the normal-state resistance.

The Ambegaokar–Baratoff relation provides a connection between the critical current and the normal-state resistance,

$$I_C = \frac{\pi\Delta(T)}{2eR_n}\tanh\left(\frac{\Delta(T)}{2k_BT}\right) \approx \frac{\pi \cdot 1.76\,k_BT_c}{2eR_n}, \tag{2.8}$$

where $\Delta(T)$ is the temperature-dependent superconducting gap. The approximation follows from the BCS theory [2] under the assumption that the measurement is performed at temperatures well below the critical temperature T_c ($\Delta(T) \approx \Delta(0)$).

Combining Eq. 2.8 with Eq. 2.6, we see that the Josephson energy can be indirectly extracted from the high-temperature resistance R_n,

$$E_J \approx \frac{\Phi_0 \cdot 1.76\, k_B T_c}{4eR_n}. \tag{2.9}$$

3. Superconducting qubit

After introducing the Josephson Junction in the previous chapter, in this chapter we explore how this element can be employed for the realization of superconducting qubits. In the literature, a wide variety of qubit designs can be found (see [3]). In this thesis, we focus on SQUIDs, which, thanks to their ability to modulate the qubit frequency, allow the implementation of fast two-qubit gates at the cost of adding one flux line per qubit. These type of qubits are used in Part III to demonstrate how to calibrate a superconducting platform with `Qibocal`.

To introduce this design, we start from the simplest qubit case, the Cooper pair box Sec. 3.1, and after discussing its main features, we move toward the transmon regime 3.2, characterized by longer coherence times. Finally, we will use the concepts developed in the previous sections to introduce SQUIDs in Sec. 3.3.

3.1. The Cooper pair box

When a small superconducting island is connected to a reservoir of Cooper pairs through a Josephson junction, the resulting device is known as a Cooper pair box (CPB), illustrated in Fig. 3.1. The system can be described in terms of the number of pairs in the island, n_I, and in the reservoir, n_R. Due to tunneling across the junction, Cooper pairs can be exchanged, and the state of the system can be represented by the excess number of pairs n on the island,

$$|n\rangle = |n_I + n,\, n_R - n\rangle\,. \tag{3.1}$$

The quantized Hamiltonian of the CPB is given by

$$\hat{H} = 4E_C\hat{n}^2 - E_J\cos(\hat{\phi}), \tag{3.2}$$

where $E_C = e^2/2C$ is the **charging energy**, determined by the capacitance C of the island. From Eq. 3.2 we see that the conjugate observables of the system are the number of Cooper pairs $\hat{n}$ and the phase difference $\hat{\phi}$.

To control the CPB, a DC voltage can be applied to the island through a gate capacitor C_g (Fig. 3.1). When a voltage V is applied, the capacitor accumulates a charge $Q_g = C_gV$, while an opposite charge $-Q_g$ appears on the qubit capacitance. Whenever the accumulated charge is a multiple of $2e$, it is energetically favorable for Cooper pairs to tunnel from the

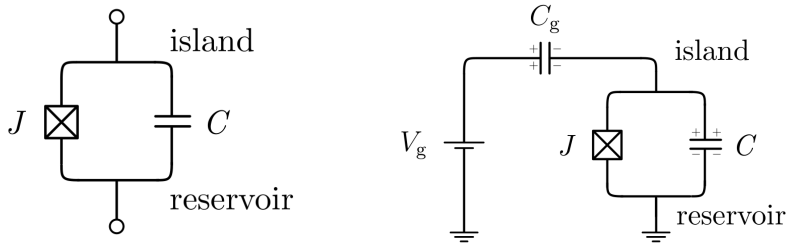


Figure 3.1.: Circuit diagram of the CPB (left) and the CPB coupled to a voltage source (right). The boxed-X element is the circuital representation of a josepshon junction. Image adapted from [1].

island to the reservoir rather than accumulate the charge on the qubit capacitor.

The Hamiltonian of the CPB in the presence of a gate voltage is

$$\hat{H} = 4E_C(\hat{n} - n_g)^2 - E_J \cos(\hat{\phi}), \tag{3.3}$$

where $n_g = C_g V/2e$ is the **dimensionless gate charge**, i.e., the charge induced on the island by the applied DC voltage.

The two computational qubit states $|0\rangle, |1\rangle$ are mapped respectively into the ground and the first excited states of the CPB. For the case $E_J/E_C = 1$, we observe that at $n_g = 0.5$ the anharmonicity $\alpha = E_{12} - E_{01}$ is large, this improves qubit control by suppressing the probability of leakage into higher energy levels, as explained in the following section 3.2. This operating point, referred to as the **sweet spot**, is particularly advantageous because the qubit frequency becomes first-order insensitive to charge noise. As a consequence, the dephasing time T_ϕ [1] [5] is enhanced, since

$$T_\varphi \propto \frac{1}{\left|\frac{dE_{01}}{dn_g}\right|}. \tag{3.4}$$

To evaluate T_φ at the sweet spot, Eq. 3.4 cannot be used directly, since the first derivative vanishes at this point. Instead, we must consider a second-derivative approximation [1],

$$T_\varphi \approx \frac{\hbar}{A_Q^2 \pi^2} \frac{1}{\left|\frac{\partial^2 E_{01}}{\partial n_g^2}\right|}, \tag{3.5}$$

where A_Q characterizes the amplitude of the charge noise.

[1] The dephasing time T_ϕ is the timescale in which the information regarding the relative phase is lost.

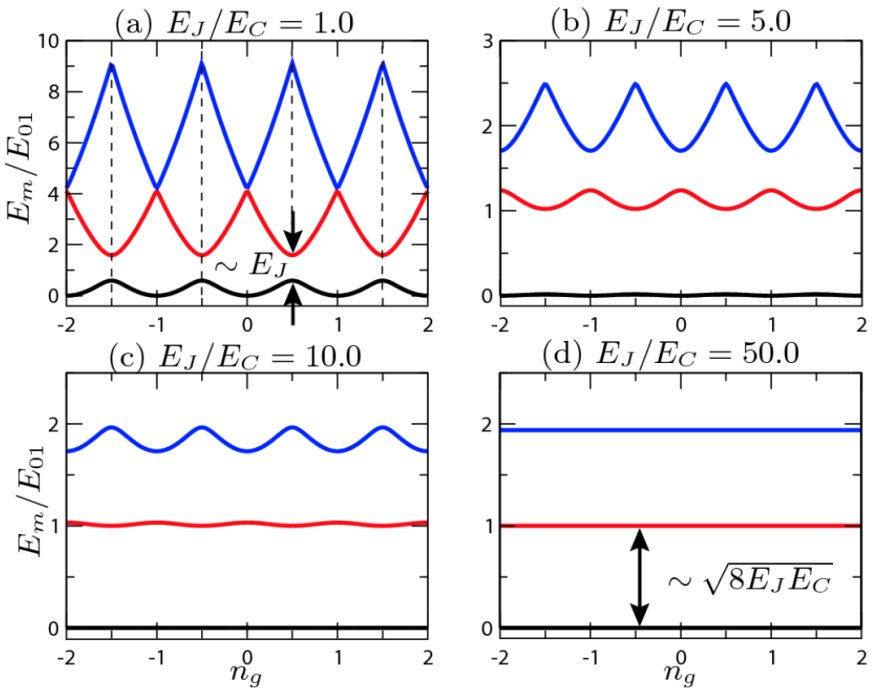


Figure 3.2.: Eigenenergies E_m of the CPB Hamiltonian for different ratios E_J/E_C (adapted from [4]).

At the sweet spot, the difference of the ground state and the first excited state E_{01} takes the form

$$E_{01}(n_g) = \sqrt{\left[4E_C(2n_g - 1)\right]^2 + E_J^2}, \tag{3.6}$$

and therefore

$$T_\varphi \approx \frac{\hbar}{A_Q^2 \pi^2} \frac{E_J}{(8E_C)^2}. \tag{3.7}$$

Using realistic values for the qubit parameters, $E_J/h = E_C/h = 6$ GHz, and for the charge noise $A_Q = 2 \times 10^{-4}\ e/\sqrt{\text{Hz}}$, we obtain an upper bound for the dephasing time T_φ on the order of 1 μs. Such a short coherence time is insufficient for practical qubit operation, this is why CPB in the regime with $E_C \sim E_J$ is not currently used in superconducting platforms.

3.2. Transmon

A Cooper pair box with $E_J/E_C \gg 1$ is called a **transmon** [4]. This design choice is motivated by the fact that increasing E_J/E_C reduces the sensitivity of the energy levels to charge noise, thereby improving the dephasing

time T_φ. As can be seen in Fig. 3.2, a higher ratio flattens the energy levels, further enhancing coherence.

Starting from Eq. 3.3, a gauge transformation can be applied to eliminate the n_g term. In the transmon regime ($E_J/E_C \gg 1$), the phase ϕ is small, allowing a Taylor expansion of the cosine potential around $\phi = 0$.

Before performing this expansion, we note that the original CPB wavefunction satisfies the periodic boundary condition

$$\psi(\phi + 2\pi) = \psi(\phi). \tag{3.8}$$

The Taylor expansion breaks this periodicity, so the boundary condition must be redefined [4] as

$$\lim_{\phi \to \infty} \psi(\phi) = 0. \tag{3.9}$$

The resulting transmon Hamiltonian becomes

$$\hat{H} = 4E_C\hat{n}^2 + \frac{E_J}{2}\hat{\phi}^2 - \frac{E_J}{4!}\hat{\phi}^4, \tag{3.10}$$

where the first two terms correspond to a harmonic oscillator and the last term introduces anharmonicity.

Using standard harmonic oscillator theory, the operators $\hat{n}$ and $\hat{\phi}$ can be expressed in terms of creation and annihilation operators $\hat{b}^\dagger, \hat{b}$:

$$\hat{\phi} = \sqrt{\xi}(\hat{b} + \hat{b}^\dagger), \tag{3.11}$$

$$\hat{n} = -\frac{i}{2\sqrt{\xi}}(\hat{b} - \hat{b}^\dagger), \tag{3.12}$$

with

$$\xi = \sqrt{\frac{2E_C}{E_J}}. \tag{3.13}$$

The transmon Hamiltonian can be expressed as

$$\hat{H} = \sqrt{8E_CE_J}\,\hat{b}^\dagger\hat{b} - \frac{E_C}{12}(\hat{b} + \hat{b}^\dagger)^4. \tag{3.14}$$

Using the rotating wave approximation, the quartic term can be simplified to

$$\hat{H} \approx \sqrt{8E_CE_J}\,\hat{b}^\dagger\hat{b} - \frac{E_C}{2}\,\hat{b}^\dagger\hat{b}^\dagger\hat{b}\hat{b}, \tag{3.15}$$

where the first term corresponds to a harmonic oscillator and the second term introduces a self-Kerr interaction, which provides the anharmonicity.

Applying perturbation theory, the energy levels of the transmon are approximately

$$E_k = \sqrt{8E_CE_J}\,k - \frac{E_C}{2}(k^2 + k), \tag{3.16}$$

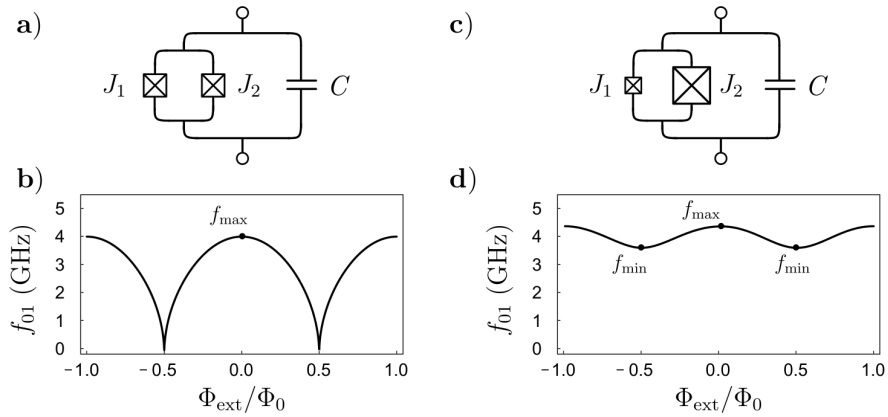


Figure 3.3.: Panels (**a-c**) show circuit representations of a SQUID qubit with different values of relative asymmetry d. When the SQUID is symmetric ($d = 1$), as in (**a-b**), the qubit frequency can be tuned over a wide range. In contrast, in the asymmetric case (**c-d**), the tunability is reduced. For executing two-qubit gates, precise control of the qubit frequency is required; therefore, slightly asymmetric SQUIDs are often preferred. Image adapted from [1].

so that the transition energy between the ground and first excited state is

$$E_{01} = \sqrt{8E_C E_J} - E_C, \tag{3.17}$$

and the anharmonicity is given by

$$\alpha = E_{12} - E_{01} = -E_C. \tag{3.18}$$

In typical transmons, the charging energy is on the order of $E_C/h \sim 200$ MHz, which is relatively small. This implies that increasing the qubit lifetime comes at the cost of a higher probability of leakage to non-computational states. However, while the anharmonicity scales as $(E_C/E_J)^{1/2}$, the decoherence time increases exponentially with $(E_J/E_C)^{1/2}$ [4] in the limit $E_J/E_C \gg 1$. Consequently, the transmon regime provides a favorable trade-off for designing superconducting qubits with long coherence times while maintaining sufficient anharmonicity for control.

3.3. SQUID

A possible enhancement for transmon qubits is the ability to tune their transition frequency. Frequency tunability allows qubits on a chip to be operated at large detunings, a fundamental property for two-qubit gate

operations. This tunability can be achieved by arranging two Josephson junctions in parallel within a superconducting loop (Fig. 3.3 a), forming a circuit commonly referred to as a **superconducting quantum interference device** (SQUID).

The Hamiltonian of a SQUID circuit with two Josephson junctions is given by

$$\hat{H} = 4E_C\hat{n}^2 - E_{J1}\cos\hat{\phi}_1 - E_{J2}\cos\hat{\phi}_2, \tag{3.19}$$

where E_{J1} and E_{J2} are the Josephson energies of the two junctions, and $\hat{\phi}_1$, $\hat{\phi}_2$ denote the corresponding phase differences across the junctions. The two phases are constrained by the quantization of magnetic flux through the SQUID loop, which enforces

$$\hat{\phi}_1 - \hat{\phi}_2 = \frac{2\pi}{\Phi_0}\phi_{ext} \pmod{2\pi}, \tag{3.20}$$

with ϕ_{ext} representing the externally applied magnetic flux.

Defining the effective phase $\hat{\phi} = (\hat{\phi}_1 - \hat{\phi}_2)/2$, the Hamiltonian reduces to

$$\hat{H} = 4E_C\hat{n}^2 - E_J(\phi_{ext})\cos\hat{\phi}, \tag{3.21}$$

where the flux-dependent Josephson energy is

$$E_J(\phi_{ext}) = (E_{J1} + E_{J2})\left|\cos\left(\pi\frac{\phi_{ext}}{\Phi_0}\right)\right|\sqrt{1 + d^2\tan^2\left(\pi\frac{\phi_{ext}}{\Phi_0}\right)}, \tag{3.22}$$

and $d = (E_{J1} - E_{J2})/(E_{J1} + E_{J2})$ quantifies the relative junction asymmetry (see Fig. 3.3).

From Eq. 3.22, the qubit transition frequency can be evaluated as

$$f_{01} = \frac{1}{\hbar}\left(\sqrt{8E_CE_J(\phi_{ext})} - E_C\right). \tag{3.23}$$

4. Circuit quantum electrodynamics

In this chapter, we address how to implement quantum gates on a qubit by coupling it to a pulse source, and how to measure its state by coupling the qubit to a resonator. This section will be particularly useful in Part III, as it provides the foundation for understanding the principles behind qubit calibration.

4.1. Qubit drive

To manipulate the state of a transmon qubit, it can be capacitively coupled to a voltage source via a gate capacitor, as already discussed for the Cooper pair box (Fig. 3.1). The Hamiltonian of the combined qubit–drive system can be written as

$$\hat{H} = 4E_C\left(\hat{n} + \frac{C_g V_d(t)}{2e}\right)^2 - E_J\cos(\hat{\phi}), \tag{4.1}$$

where $V_d(t)$ represents a generic oscillating voltage pulse of the form

$$V_d(t) = A\,\varepsilon(t)\sin(\omega_d t + \alpha), \tag{4.2}$$

where A is the pulse amplitude,ω_d is the drive frequency, α is the phase of the pulse and $\varepsilon(t)$ is the modulation of the pulse. Neglecting constant terms, the Hamiltonian can be separated into the qubit part and the drive part:

$$\hat{H} = 4E_C\hat{n}^2 - E_J\cos(\hat{\phi}) + 2e\frac{C_g V_d(t)}{C_\Sigma}\hat{n}, \tag{4.3}$$

with $E_C = e^2/2C_\Sigma$ and $C_\Sigma = C_g + C$. Using the expressions for $\hat{n}$ in terms of ladder operators (Eq. 3.12), the drive term becomes

$$\hat{H}_d = 2e\frac{C_g V_d(t)}{C_\Sigma}\hat{n} = -i\frac{e}{\sqrt{\xi}}\frac{C_g}{C_\Sigma}V_d(t)(\hat{b} - \hat{b}^\dagger). \tag{4.4}$$

Restricting the system to the computational $\{|0\rangle, |1\rangle\}$ subspace, the Hamiltonian reduces to

$$\hat{H} = -\frac{\hbar\omega_q}{2}\hat{\sigma}_Z + \hbar\Omega\,\varepsilon(t)\sin(\omega_d t + \alpha)\hat{\sigma}_Y, \tag{4.5}$$

where $\hat{\sigma}_Z$, $\hat{\sigma}_Y$ are the Pauli operators and the Rabi frequency Ω is given by

$$\Omega = \frac{e}{\hbar\sqrt{\xi}}\frac{C_g}{C_\Sigma}A, \tag{4.6}$$

and is directly proportional to the drive amplitude A.

When the drive frequency is near resonance with the qubit, $\omega_d \approx \omega_q$, the drive Hamiltonian in Eq. 4.4 can be expressed in the qubit frame and simplified using the rotating wave approximation (RWA):

$$\hat{H} = -\frac{\hbar(\omega_q - \omega_d)}{2}\hat{\sigma}_Z + \frac{\hbar\Omega}{2}\varepsilon(t)\big(\sin\alpha\,\hat{\sigma}_Y + \cos\alpha\,\hat{\sigma}_X\big), \tag{4.7}$$

where the phase α of the drive determines the axis of rotation on the Bloch sphere.

In particular, for $\alpha = \pi/2$, the drive term reduces to

$$\hat{H}'_d(t) = \frac{\hbar\Omega}{2}\varepsilon(t)\,\hat{\sigma}_Y, \tag{4.8}$$

which corresponds to a rotation about the y-axis. The evolution of the qubit state under this pulse is given by

$$|\psi\rangle = \exp\left(-\frac{i}{\hbar}\int_0^\infty \hat{H}'_d(t')\,dt'\right)|0\rangle = \exp\left(-\frac{i\theta}{2}\hat{\sigma}_Y\right)|0\rangle, \tag{4.9}$$

where the rotation angle θ is proportional to the time-integrated amplitude of the pulse:

$$\theta = \Omega\int_0^\infty \varepsilon(t')\,dt'. \tag{4.10}$$

The microwave drive signal $V_d(t)$ can be conveniently decomposed into its in-phase and quadrature components as

$$\begin{aligned} V_d(t) &= A\,\varepsilon(t)\big(\cos\alpha\,\sin(\omega_d t) + \sin\alpha\,\cos(\omega_d t)\big) \\ &= I(t)\,\sin(\omega_d t) + Q(t)\,\cos(\omega_d t), \end{aligned} \tag{4.11}$$

where $I(t)$ and $Q(t)$ denote the in-phase and quadrature components, respectively. In terms of these components, the drive Hamiltonian in the rotating frame (Eq. 4.7) can be expressed as

$$\hat{H}_d(t) = \frac{\hbar\Omega}{2}\varepsilon(t)\big(I(t)\,\hat{\sigma}_Y + Q(t)\,\hat{\sigma}_X\big). \tag{4.12}$$

By appropriately shaping and controlling the I and Q signals using an **arbitrary waveform generator** (AWG), one can perform arbitrary rotations of the qubit state on the Bloch sphere.

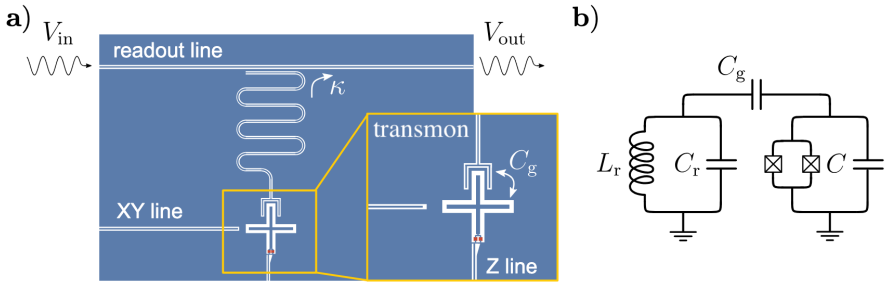


Figure 4.1.: (a) Schematic of a tunable transmon qubit capacitively coupled to a coplanar waveguide resonator. The resonator is inductively coupled to a readout line. The transmon is also connected to an XY control line (drive line) for qubit state manipulation and a Z control line (flux line) for tuning the qubit frequency. (b) Equivalent circuit representation of the tunable transmon coupled to the microwave resonator.

4.2. Qubit readout

The measurement of a transmon qubit is typically performed via its interaction with a microwave resonator, as in Fig. 4.1, a technique commonly referred to as **dispersive readout** [6]. In this regime, the qubit-resonator coupling induces a state-dependent shift in the resonator frequency, such that the resonator frequency takes the value $\omega_{r,0}$ when the qubit is in the ground state $|0\rangle$, and $\omega_{r,1}$ when the qubit is in the excited state $|1\rangle$. By applying a microwave probe at a frequency intermediate between $\omega_{r,0}$ and $\omega_{r,1}$, the transmitted or reflected signal acquires a phase shift that encodes information about the qubit state. This phase shift can then be detected and processed to perform a high-fidelity, single-shot measurement of the qubit.

Let's consider a qubit capacitively coupled to a resonator

$$\hat{H} = \hbar\omega_r \hat{a}^\dagger \hat{a} + 4E_C \hat{n}^2 - E_J \cos\hat{\phi} + \frac{4E_C}{e}\hat{n} C_g \hat{V}, \tag{4.13}$$

with $\hat{V} = \hat{Q}/C_r$ the voltage across the resonators. We can express the Hamiltonian in terms of the ladder operators

$$\hat{H} = \hbar\omega_r \hat{a}^\dagger \hat{a} + \sqrt{8E_C E_J}\hat{b}^\dagger \hat{b} - \frac{E_c}{12}(\hat{b} + \hat{b}^\dagger)^4 + \hbar g(\hat{b}^\dagger - \hat{b})(\hat{a} - \hat{a}^\dagger), \tag{4.14}$$

where we introduced the coupling strength:

$$g = \frac{E_C}{\hbar e}\left(\frac{E_J}{2E_C}\right)^{1/4} \frac{C_g}{C_r}\sqrt{2\hbar\omega_r C_r}. \tag{4.15}$$

In the regime where the qubit and resonator frequencies are much larger than the coupling strength, $\omega_r, \omega_q \gg g$, the Hamiltonian of Eq. 4.14 can be simplified using the rotating wave approximation (RWA):

$$\hat{H} = \hbar\omega_r \hat{a}^\dagger \hat{a} + \sqrt{8E_C E_J}\,\hat{b}^\dagger \hat{b} - \frac{E_C}{12}(\hat{b} + \hat{b}^\dagger)^4 + \hbar g(\hat{b}^\dagger \hat{a} - \hat{b}\hat{a}^\dagger). \tag{4.16}$$

By further restricting the transmon to its two lowest energy levels, the system reduces to the well-known **Jaynes-Cummings Hamiltonian** [6]:

$$\hat{H} = \hbar\omega_r \hat{a}^\dagger \hat{a} - \frac{\hbar\omega_{01}}{2}\hat{\sigma}_Z + \hbar g\left(\hat{\sigma}^+ \hat{a} - \hat{\sigma}^- \hat{a}^\dagger\right), \tag{4.17}$$

where ω_{01} is the qubit transition frequency, and $\hat{\sigma}^\pm$ are the qubit raising and lowering operators.

This Hamiltonian forms the basis for the dispersive interaction between the qubit and the resonator, which underlies the standard microwave readout technique.

When the detuning between the qubit and the resonator is large compared to the coupling, $\Delta = |\omega_r - \omega_q| \gg g$, the system enters the **dispersive regime**. In this regime, the Hamiltonian can be further simplified to

$$\hat{H}_{\text{disp}} = \hbar(\omega_r - \chi\hat{\sigma}_Z)\hat{a}^\dagger \hat{a} - \frac{\hbar}{2}(\omega_{01} + \chi)\hat{\sigma}_Z, \tag{4.18}$$

where $\chi = g^2/\Delta$ is the **dispersive shift**.

A more rigorous condition [7] for the dispersive regime can be derived from the eigenenergies of the Jaynes-Cummings Hamiltonian:

$$E_{n\pm} = \hbar\left(n - \frac{1}{2}\right)\omega_r \pm \frac{\hbar|\Delta|}{2}\sqrt{1 + \frac{4\chi n}{\Delta}}. \tag{4.19}$$

Perturbation theory is valid when $\frac{2g}{\Delta}\sqrt{n} \ll 1$, which can be simplified to $g/\Delta \ll 1$. For a given g/Δ, the maximum photon number for which this approximation holds is

$$n_{\text{crit}} = \left(\frac{\Delta}{2g}\right)^2. \tag{4.20}$$

Taking into account higher energy levels of the transmon, the dispersive shift is modified to

$$\chi = \frac{g^2}{\Delta\left(1 + \frac{\Delta}{\alpha}\right)}, \tag{4.21}$$

where α denotes the qubit anharmonicity.

From Eq. 4.18, it is evident that the resonator frequency depends on the qubit state. When the qubit is in the ground state $|0\rangle$, the resonator frequency is

$$\omega_{r,0} = \omega_r - \chi,$$

while for the first excited state $|1\rangle$, the resonator frequency becomes

$$\omega_{r,1} = \omega_r + \chi.$$

Equation 4.18 can also be expressed in the form

$$\hat{H} = \hbar\omega_r \hat{a}^\dagger \hat{a} - \frac{\hbar}{2}\left(\omega_{01} - \chi + 2\chi\, \hat{a}^\dagger \hat{a}\right)\hat{\sigma}_Z, \tag{4.22}$$

which highlights that the qubit-resonator coupling shifts the qubit transition frequency by χ, commonly referred to as the **Lamb shift**. Furthermore, the qubit frequency depends on the number of photons in the resonator: the addition of a single photon shifts the qubit frequency by 2χ, a phenomenon known as the **AC Stark shift**.

5. Qubit and environment

One of the most challenging aspects to control when working with a superconducting platform is its coupling to the external environment. In order to operate a qubit, some degree of coupling with various instruments is necessary; however, this often results in greater exposure of the qubit to environmental noise, leading to loss of coherence. To better understand this phenomenon, in Sec. 5.1 we will briefly discuss the effects of the interaction between the qubit and its environment, together with the definition of coherence times. In Sec. 5.2 , we will address how the qubit–resonator–environment coupling influences the T_1.

5.1. Coherence times

The Bloch-Redfield model [1] provides a widely used theoretical framework for describing the decoherence of a two-level system (qubit) weakly coupled to its environment. Its central assumption is that the environmental noise has correlation times much shorter than the timescales of the qubit dynamics, implying that the environment is effectively **memoryless** (Markovian). This model is particularly suited for analyzing decoherence in systems such as superconducting quantum circuits.

Within the Bloch-Redfield formalism, qubit relaxation is characterized by two fundamental rates [5]:

1. **Longitudinal Relaxation Rate (Γ_1):** Defined as $\Gamma_1 = 1/T_1$, this rate governs the decay of energy from the qubit. It describes the depolarization along the qubit's quantization axis (the z-axis of the Bloch sphere), driving the population toward the ground state.

2. **Transverse Relaxation Rate (Γ_2):** Defined as $\Gamma_2 = 1/T_2$, this rate governs the decay of quantum coherence for a superposition state. It can be decomposed as

$$\Gamma_2 = \frac{\Gamma_1}{2} + \Gamma_\phi, \tag{5.1}$$

   where Γ_ϕ is the **pure dephasing rate**, arising from longitudinal noise (along z) that stochastically modulates the qubit frequency. Such fluctuations induce random phase shifts, leading to a loss of coherence in the x-y plane of the Bloch sphere without any energy exchange.

For an initial qubit state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{5.2}$$

the Bloch-Redfield model predicts the time evolution of the density matrix as

$$\rho_{\mathrm{BR}}(t) = \begin{pmatrix} 1-(1-|\alpha|^2)e^{-\Gamma_1 t} & \alpha\beta^* e^{i\delta\omega t}e^{-\Gamma_2 t} \\ \alpha^*\beta e^{-i\delta\omega t}e^{-\Gamma_2 t} & |\beta|^2 e^{-\Gamma_1 t} \end{pmatrix}. \tag{5.3}$$

This expression shows that the diagonal elements decay exponentially with a characteristic time $T_1 = 1/\Gamma_1$, representing energy relaxation toward the ground state, while the off-diagonal elements decay with a characteristic time $T_2 = 1/\Gamma_2$, capturing the loss of quantum coherence. The factors $e^{\pm i\delta\omega t}$ account for coherent phase accumulation in the rotating frame, where $\delta\omega = \omega_q - \omega_d$ is the detuning between the qubit transition and the drive frequency. It is important to note that the additive relation $\Gamma_2 = \Gamma_1/2 + \Gamma_\phi$ and the resulting exponential decay of coherence are strictly valid only for noise sources with a **Lorentzian spectrum** (e.g., white noise) centered at zero frequency; more complex or low-frequency noise spectra may require a more general treatment.

A significant limitation of the standard Bloch-Redfield model arises when modeling the dominant noise sources in superconducting qubits, such as flux, charge, and critical-current noise, which typically exhibit a $1/f$**-type power spectral density**, i.e., the power spectral density is inversely proportional to the frequency of the signal. Noise of this type is particularly problematic because its spectrum diverges at low frequencies ($\omega \to 0$), indicating the presence of slow processes with long correlation times. This directly violates the model's fundamental assumption of short correlation times, leading to two main consequences: the decay of the off-diagonal coherence terms becomes **non-exponential**, and the additive rate relation $\Gamma_2 = \Gamma_1/2 + \Gamma_\phi$ **breaks down**.

To accurately model dephasing induced by $1/f$ noise, the density matrix description must be modified. A standard approach is to replace the exponential dephasing factor $\exp(-\Gamma_\phi t)$ with a **Gaussian decay envelope**, $\exp\left[-(t/T_{\phi,G})^2\right]$. This Gaussian contribution is typically treated as separable from the T_1-induced exponential decay, resulting in an overall coherence decay of the form

$$e^{-\Gamma_1 t/2} \cdot e^{-(t/T_{\phi,G})^2}. \tag{5.4}$$

5.2. Purcell effect

Let us consider a qubit coupled to a resonator, which in turn is coupled to its environment. The total Hamiltonian of the system can be written as

$$\hat{H} = \hat{H}_q + \hat{H}_r + \hat{H}_e + \hat{H}_{r,q} + \hat{H}_{r,e}, \tag{5.5}$$

where $\hat{H}_q$ and $\hat{H}_r$ describe the qubit and the resonator, respectively, and $\hat{H}_e$ represents the environment. The environment is modeled as an infinite set of harmonic oscillators:

$$\hat{H}_e = \sum_j \hbar\omega_j \hat{c}_j^\dagger \hat{c}_j, \tag{5.6}$$

and its interaction with the resonator is described by

$$\hat{H}_{r,e} = \hbar \sum_j \lambda_j \left(\hat{a}\hat{c}_j^\dagger + \hat{a}^\dagger \hat{c}_j \right), \tag{5.7}$$

where λ_j quantifies the coupling strength between the resonator mode $\hat{a}$ and the j-th environmental oscillator $\hat{c}_j$.

Assuming the qubit is initially in the excited state $|e\rangle$ with a small photonic component arising from the Jaynes-Cummings interaction in the dispersive regime, the initial state of the global system can be written as

$$|i\rangle = |e, 0, 0\rangle + \frac{g}{\Delta} |g, 1, 0\rangle , \tag{5.8}$$

where the state vector is ordered as qubit, resonator, and environment. Considering a final state

$$|f\rangle = |g, 0, 1_k\rangle \tag{5.9}$$

with $E_f = E_i$, the decay rate can be computed using Fermi's golden rule:

$$\frac{1}{T_1} = \frac{2\pi}{\hbar^2} p(\omega_k) \left| \langle f | \hat{H}_{r,e} | i \rangle \right|^2 , \tag{5.10}$$

where $p(\omega_k)$ is the density of states of the environment at frequency ω_k. Evaluating the matrix element and summing over the bath modes yields the Purcell-limited relaxation time of the qubit:

$$T_1 = \frac{\Delta^2}{\kappa g^2}, \tag{5.11}$$

with $\kappa = 2\pi\, p(\omega_k) \lambda_k^2$ representing the decay rate of the resonator into the environment.

Equation 5.11 defines the **Purcell limit** on the qubit relaxation time T_1. This limit increases with the detuning Δ and decreases with the resonator linewidth κ. In practice, these dependencies lead to a trade-off: a small κ enhances T_1, improving qubit coherence, while a larger κ is desirable to allow photons to leave the resonator quickly, enabling fast measurement. This apparent contradiction is resolved through the use of **Purcell filters**, which are microwave filters placed between the resonator and the readout line. Purcell filters allow rapid release of measurement photons while simultaneously protecting the qubit from decay, thereby increasing the effective coherence time.

Part II.

Full stack quantum computing

6. Project overview

6.1. Introduction

The practical realization of quantum computers for executing quantum algorithms requires overcoming a series of complex, interdependent challenges across several technical domains. At the most fundamental level, advanced fabrication techniques are essential for producing qubit chips with longer coherence times and minimized crosstalk. But at the same time, reliable quantum computation extends far beyond chip manufacturing. A superconducting quantum processor works within an ultra-low-temperature cryogenic environment, placed in a carefully engineered cryostat that suppresses thermal noise while maintaining links to room-temperature control electronics, which generate and shape the microwave pulses needed for qubit operations. To preserve signal integrity and protect fragile quantum states, a sophisticated attenuation and filtering scheme is indispensable for suppressing thermal photons and other sources of disruption.

Once the hardware is in place, reliable software is needed to coordinate the instruments that control the qubits. This software must include tools for calibration, a process where the chip's parameters are repeatedly measured and fine-tuned to ensure high-fidelity qubit operations. The main difficulties in this process come from temporal drifts and environmental fluctuations, which make continuous monitoring and adjustment of the platform essential.

When a quantum circuit is executed on a calibrated quantum hardare two main steps are performed: the first one is the transpilation, where each gate of the circuit is decomposed into the native gates set of the specific platform and then the circuit is rewritten to follow the chip topology by strategically adding SWAP gates that swaps the states of two qubits, this is followed by the compilation where the transpiled circuit is translated in pulse sequence.

In conclusion, a superconducting quantum computer can divided in four layers: the **hardware layer**, collecting both the quantum chip and the cryogenic setup, the **pulse layer**, comprising a compiler that translates gates into precisely timed control waveforms; the **calibration layer**, for the maintainance and optimization of qubit parameters and gate fidelities; the **circuit layer**, featuring a transpiler that modifies quantum algorithms in accordance with hardware constraints. This layered methodology facilitates specialized optimizations at each stage while ensuring smooth integration across the quantum computing stack.

The qibo framework [8, 9, 10] presented in this thesis will serve as an unifying framework that integrates all these layers.

6.2. Idea and Motivations

The aim of our project is to develop a comprehensive full-stack software framework for quantum computers, capable of handling all levels of abstraction, from the low-level control of pulses to the high-level design of quantum logical circuits. This software is more than just a technical tool; it represents a collective effort to build a collaborative network of small- and medium-sized quantum laboratories. By sharing expertise and maintenance effort, these labs can work together on a community-driven initiative that accelerates research and promotes open innovation in the field of quantum computing.

As seen in the previous section, a reliable software infrastructure is essential because operating a quantum computer involves coordinating several complex and interdependent processes. Tasks such as gate calibration, circuit transpilation and compilation, and the execution of pulse sequences require careful orchestration and precision. These operations must be executed on top of control system APIs (application programming interfaces) in a consistent and dependable manner. Without proper automation and integration, the preparation and execution of experiments become labor-intensive, error-prone, and inefficient. Developing solid software libraries is therefore crucial to streamline these processes and to enable researchers to use quantum devices with greater ease and reliability.

To be truly effective, the software must be **open source**, allowing the community to contribute, improve, and adapt it to their needs, **platform independent**, ensuring compatibility with a wide range of hardware setups across different laboratories. Finally, the software should be built around a **multi-layered architecture** that clearly separates low-level control tasks from high-level algorithm execution. This layered approach enables flexibility and supports a wide variety of use cases, from running quantum algorithms to performing calibration and control experiments.

By uniting these principles under a single software framework, we aim to provide a powerful, adaptable, and collaborative foundation for advancing quantum computing research.

6.3. Software design

In Fig. 6.1, we provide a schematic representation of the architectural layout of the `Qibo` ecosystem. This framework is logically divided into two principal components: the language API, which serves as the user-facing interface for algorithm design, and the backend implementations,

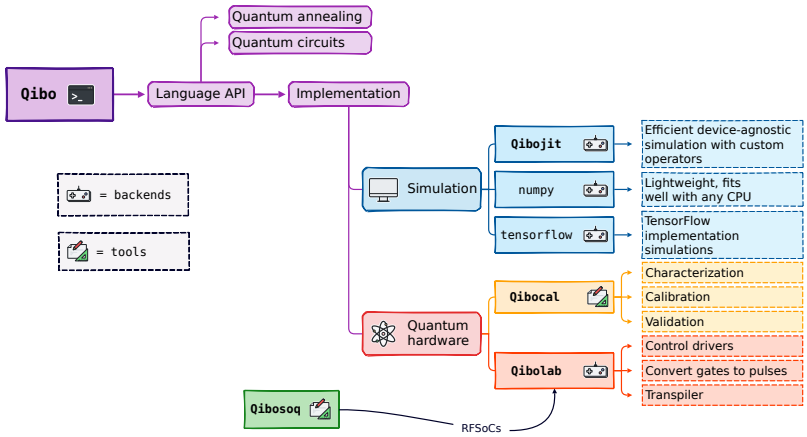


Figure 6.1.: Schematic overview of `Qibo` software components, including backends and tools.

which enable the execution of quantum algorithms on a range of classical and quantum hardware platforms.

The language API comprises a suite of high-level abstractions designed to facilitate the rapid prototyping and development of quantum algorithms. It supports multiple computational paradigms, including the circuit-based and adiabatic models, and employs Python as its primary programming language to promote accessibility and ease of use. This API layer is designed to be expressive and modular, providing an intuitive interface for both novice users and experienced researchers.

Within the circuit model, the API offers a comprehensive set of primitives for the exact manipulation of quantum states. These include fundamental operations such as single-qubit and two-qubit gates, as well as more complex constructs like Toffoli gate (three qubit gate) and gate fusion strategies. The API further incorporates an extensive measurement interface, enabling users to extract computational results through sampling (i.e., shots) with a high degree of flexibility and control.

Beyond basic circuit construction and execution, `Qibo` integrates a collection of variational optimization tools, including optimizers and callback mechanisms tailored to variational quantum algorithms (VQAs). Additionally, a dedicated module provides access to a wide array of quantum information primitives, allowing users to conduct theoretical and applied studies in quantum computation and information science within a unified software environment.

From an implementation point of view, `Qibo` is architected to support multiple execution backends, which are responsible for translating high-level quantum programs into executable on specific hardware or

simulation platforms. Each backend adheres to a standardized abstract interface, which prescribes the set of methods required to enable full compatibility with the language API. This modular backend design ensures extensibility and facilitates the integration of new computational targets without modifications to the core API.

The framework supports various simulation backends optimized for classical hardware, enabling efficient testing and validation of quantum algorithms. Moreover, through the integration with `Qibolab`, the same high-level codebase can be executed directly on quantum hardware platforms. `Qibolab` effectively acts as a hardware backend, dynamically adapting to the chosen experimental setup, once a specific quantum device is selected by the user.

`Qibolab` provides the essential middleware required to abstract the complexities of hardware control and integration. It enables the execution of circuit-based quantum algorithms on custom, self-hosted quantum processing units. By encapsulating the low-level control logic, `Qibolab` allows experimentalists and developers to focus on algorithmic design, while ensuring standardized, extensible, and hardware-agnostic interaction with the quantum device. The first officially supported physical implementation is based on superconducting qubits, reflecting current trends in quantum hardware development.

To facilitate pulse-level control and execution, `Qibosoq` [11] was introduced as an open-source server-side software suite tailored for radio-frequency system-on-chip (RFSoC) platforms. It enables the deployment of arbitrary pulse sequences and quantum algorithms on self-hosted quantum processors using exclusively open-source components. `Qibosoq` bridges the firmware provided by Qick [12] (Quantum Instrumentation Control Kit) with the `Qibo` framework, thereby supporting both low-level experimental operations and higher-level gate-based applications within a single coherent stack.

Complementing these capabilities, the calibration layer is addressed by `Qibocal`, a platform-agnostic suite of calibration routines built upon the abstractions provided by `Qibolab`. `Qibocal` enables automated and reproducible calibration workflows across heterogeneous hardware platforms, ensuring optimal performance and reliability of quantum operations.

The modular and extensible nature of the `Qibo` ecosystem also opens pathways for the development of additional tooling and application domains. These include, but are not limited to, quantum chemistry simulations, advanced multi-qubit calibration protocols, benchmarking suites, and hybrid quantum-classical algorithms inspired by machine learning. Furthermore, the architecture supports the future integration of additional backends, whether simulation-based or interfaced with novel quantum hardware platforms, thus reinforcing `Qibo` 's role as a foundational middleware for the evolving landscape of quantum computing.

In the next three chapters, the main libraries developed within the ecosystem, Qibo, Qibolab, and Qibocal, will be described in detail. Re-

garding the latter two, not only will a general description be provided, but we will also examine the details of their software and explore their applications in the context of managing a quantum laboratory.

7. Qibo

7.1. Qibo stack

Simulation plays a vital role in quantum computing research, especially during the current Noisy Intermediate-Scale Quantum (NISQ) era [13]. In this context, accurate simulation results are essential for validating quantum algorithms and developing error mitigation strategies.

`Qibo` supports simulation of both gate-based and adiabatic quantum computing models on classical hardware. Its modular architecture enables quantum algorithms to run on three distinct simulation backends, each optimized for different use cases, as illustrated in Fig. 6.1.

Fig. 7.1 illustrates the schematic structure of the `Qibo` framework. At the bottom lies the abstraction layer, where quantum circuits and gates are defined. Built on top of this layer are backend implementations using TensorFlow [14] and NumPy primitives [15], which enable the simulation of quantum algorithms such as the Variational Quantum Eigensolver (VQE) and the execution of measurement shots.

These algorithms are designed to be backend-agnostic, allowing smooth switching between different simulation engines. In addition, many models supported by `Qibo` (VQE, QAOA, and adiabatic evolution) rely on external optimization libraries: TensorFlow for stochastic gradient descent and SciPy [16] for quasi-Newton methods.

To simplify usage, `Qibo` exposes a high-level Python API, providing a user-friendly entry point for building and executing quantum algorithms.

7.2. Simulation Backends

This section provides an overview of the strengths and limitations of the three available `Qibo` backends (`numpy`, `tensorflow`, and `qibojit`) and offers guidance on selecting the most appropriate backend for specific applications.

The `numpy` backend, built on NumPy primitives as detailed in [17], is a lightweight option that supports single-threaded CPU simulations with moderate performance. It is generally suitable for quantum circuits with up to 20 qubits. One of its key strengths is broad compatibility across various classical system architectures. This makes it a stable and reliable choice, particularly in development environments such as laboratories where quantum platforms are being deployed and tested.

The second backend is based on TensorFlow [14] primitives. Similarly to the `numpy` one, it can be used for solving problems involving a limited

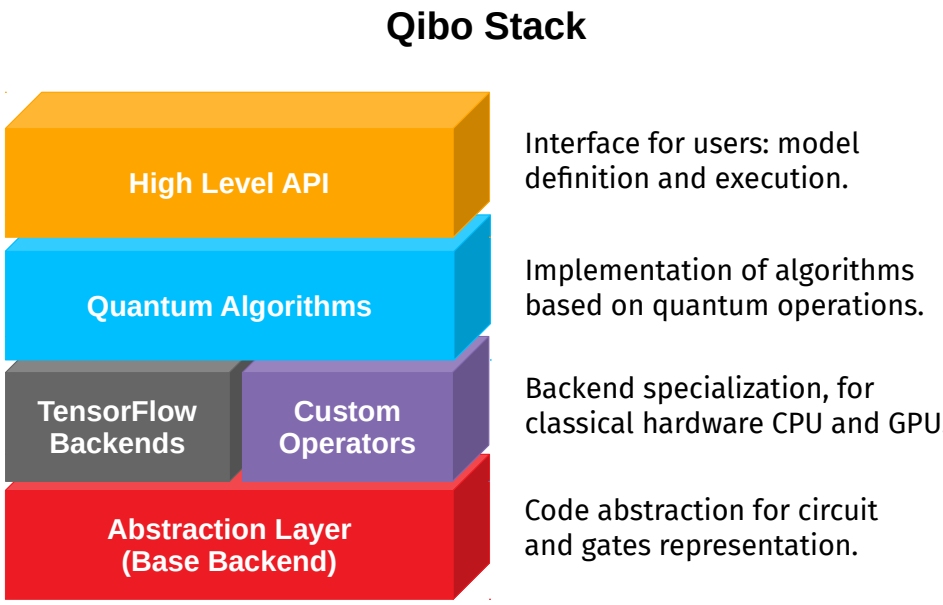


Figure 7.1.: Schematic view of the `Qibo` structure design.

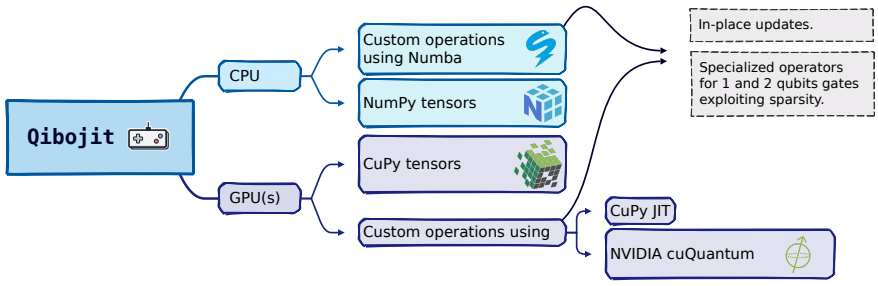


Figure 7.2.: Schematic description of the `qibojit` backend features.For CPU `qibojit` provides operators in Python using Numba's `njit` decorator. Regarding GPU, we rely on Cupy, specifically the `Rawkernel` method. We have also integrated the NVIDIA quantum simulation library *cuQuantum*.

number of qubits, although it allows to perform quantum simulation on multi-threading CPU and single-GPU.

The `tensorflow` backend leverages TensorFlow's advanced optimization capabilities, including state-of-the-art gradient-based optimizers. This is especially advantageous for Quantum Machine Learning (QML) applications, where automatic differentiation can be used to efficiently train hybrid quantum-classical models [18].

In `Qibo` optimization module, we provide the `qibo.optimizers.sgd` function, which utilizes TensorFlow's automatic differentiation to perform gradient-based optimization. This function is fully customizable, allowing developers to tailor it to their specific needs while benefiting from the underlying features of TensorFlow. Note that using this function requires selecting the `tensorflow` backend.

In Machine Learning, gradients are typically computed using the Back-Propagation (BP) algorithm [19], which involves storing intermediate copies of matrices and vectors throughout the process. Since TensorFlow relies on this approach, the `tensorflow` backend requires duplicating the state vector during simulation. This results in higher memory usage and reduced computational efficiency.

The third backend, `qibojit` [20], is a high-performance simulator that leverages Just-In-Time (JIT) compilation and custom operators for efficient state vector manipulation. It optimizes the application of quantum gates by exploiting matrix properties such as sparsity and symmetry, and by modifying data structures in-place rather than creating new copies. The architecture of `qibojit`, including its implementations for both CPU and GPU environments, is illustrated in Fig. 7.2.

The `qibojit` backend supports multi-threaded CPU execution, as well as GPU and multi-GPU configurations. CPU-based simulations are implemented using NumPy tensors and accelerated with Numba [21]. For GPU and multi-GPU setups, we use CuPy [22] as the underlying tensor library.

Two distinct GPU acceleration strategies are employed. The first leverages CuPy's `RawKernel` interface, which allows writing custom CUDA kernels in C++ and integrating them directly into Python code. The second utilizes high-performance primitives provided by NVIDIA's cuQuantum SDK [23]. Given its efficiency and scalability, `qibojit` is the recommended backend for simulating systems with a large number of qubits.

Despite ongoing advances in quantum hardware, simulation remains a vital tool for development and validation. To further enhance `Qibo` 's simulation capabilities, we are actively developing new backends. These include support for multi-node distributed simulations of state vectors, as well as a fundamentally different simulation approach based on tensor networks [24, 25, 26, 27, 28].

8. Qibolab

8.1. Introduction

The successful deployment of quantum computing algorithms relies on both quantum hardware and middleware software tailored to control instruments specific to each quantum platform.

The purpose of middleware is to offer standardized software tools that abstract the complexity of heterogeneous software interfaces. This abstraction bridges high-level quantum algorithms based on the quantum circuit model and low-level driver commands specific to individual experimental setups and instrumentation.

In this chapter, we introduce `Qibolab` [9], a software library that extends `Qibo`'s capabilities to run quantum algorithms on self-hosted quantum hardware platforms. `Qibolab` provides a dedicated application programming interface (API) that supports quantum circuit design, qubit calibration, and instrument control via arbitrary pulse sequences. It also enables low-level driver operations such as parameter sweepers and transpilation of circuits to match the native gate set and topology of the target quantum platform.

This chapter begins with a description of the project's status, design, and modules in Section 8.2. Section 8.3 then provides a detailed overview of the `Qibolab` library (v0.1.0), followed by application examples for superconducting qubit platforms in Section 8.4. Finally, Sec. 8.5 presents the detail of the emulator platform.

8.2. Quantum hardware support

Quantum computers can be built using a variety of physical systems, including superconducting circuits [29], trapped ions [30], and neutral atoms [31], among others.

While this thesis focuses on superconducting devices, `Qibolab` is designed as an extensible abstraction library that can support other quantum technologies as well. The only requirement is that the experimental setup consists of instruments capable of communicating with each other and with the quantum processing units (QPUs). As detailed in Sect. 8.3, any experimental configuration can be represented by subclassing the `Platform` class and implementing the appropriate `Instruments` and `Qubits` methods, with all connections defined via the `Channels` class.

As illustrated in Fig. 6.1, `Qibolab` provides all the essential components

required to build a backend for executing quantum algorithms on self-hosted quantum processing units (QPUs).

This integration is made seamless by `Qibo`'s modular architecture, which allows users to implement custom backends with minimal effort. In the case of a *hardware* backend, this modularity enables developers to concentrate solely on the driver level.

`Qibolab` provides an API for defining `Pulse` objects, enabling low-level control over microwave pulse sequences similar to other frameworks [32, 33, 34, 35]. This interface simplifies the implementation of both experiments and calibration protocols, offering a practical abstraction given the varying waveform definitions across different instruments.

Another key component shown in Fig. 6.1 is the set of drivers that connect `Qibo` to various instruments. To generate the microwave pulses required for quantum gate operations, the system typically employs Arbitrary Waveform Generators (AWGs), Digital-to-Analog Converters (DACs), and Analog-to-Digital Converters (ADCs) often integrated within Field Programmable Gate Arrays (FPGAs).

These devices typically come with their own control libraries or packages, such as Qblox Instruments [36], Qcodes [37], and LabOneQ [34]. Despite this heterogeneity, `Qibolab` defines a unified interface that standardizes access to the essential methods needed to control QPUs across different hardware platforms.

Finally, `Qibolab` handles all the necessary steps to prepare a fully characterized quantum device for circuit execution. This includes a compilation stage that translates quantum gates into low-level pulse instructions.

Figure 8.1 illustrates a basic laboratory setup for QPU control. `Qibolab` runs on a host computer, which typically communicates with the control electronics via a network protocol. These electronics generate the required pulses and are connected to the QPU through various channels: readout and feedback channels (used in a closed loop for qubit measurement), drive channels (for applying quantum gates), and, in the case of flux-tunable qubits, flux channels (for adjusting qubit frequencies).

8.3. Software abstractions

`Qibolab` offers a cohesive framework for managing the various electronics required to operate a quantum computer. To accomplish this, we offer software abstractions and patterns that laboratories can adopt to manage their self-hosted devices. We provide drivers for various commercial instruments as a use case, which we deploy to demonstrate the library and present benchmarks in Sec. 8.4. The subsequent sections provide a detailed description of the software abstractions and the supporting drivers.

`Qibolab` offers two primary interface objects: the `Pulse` object for specifying arbitrary pulses to be applied to qubits, and the `Platform`,

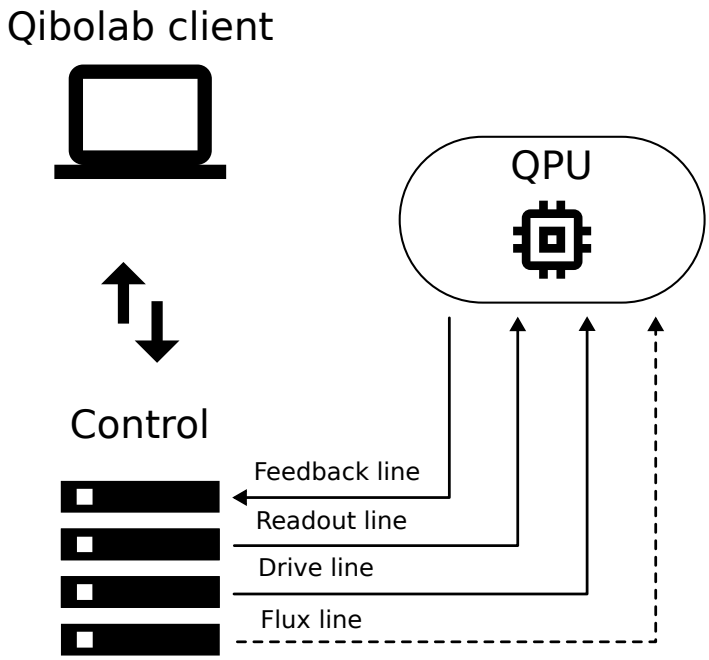


Figure 8.1.: Basic setup of a self-hosted QPU. The host computer running `Qibolab` communicates with the different electronics used to control a QPU.

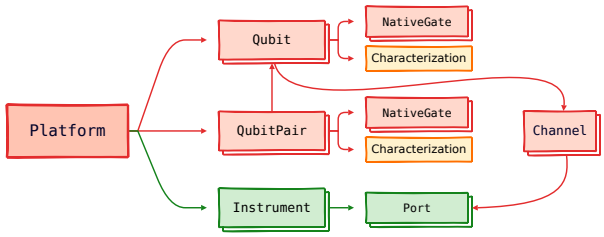


Figure 8.2.: Hierarchy of objects inside the `Platform`.

which supports the execution of these pulses on a designated QPU.

Pulses are the fundamental components of programs executed on quantum platforms. They can be used to read out the state of a qubit, induce a change in its state, or manipulate a qubit resonance frequency in order to enable two-qubit interactions. Qibolab supports pulse objects for each operational mode, with each Pulse object containing data on amplitude, frequency, phase, start time, and length, essential for the generation of physical pulses. We additionally provide the capability to generate waveforms of various shapes, including Rectangular, Gaussian, or DRAG [38].

Experiments on QPUs require the application of multiple pulses across various qubits. In Qibolab, Pulses may be aggregated within a PulseSequence. The Pulse API offers the flexibility necessary to schedule such sequences by specifying the start time of each individual pulse and allowing for overlapping pulses. This feature is essential for readout multiplexing [39].

Abstract sequences of pulses defined with the Pulse API are available on hardware via a Platform. It orchestrates the various instruments for qubit control. Each Platform instance is associated with a particular quantum chip that is managed by a designated set of instruments. It enables users to execute a single sequence, a batch of sequences, or complete a sweep, during which one or more pulse parameters are updated in real-time within the control instrument. Executing sequences in batches or performing real-time sweeps significantly enhances the speed of qubit calibration and characterization procedures.

The Platform consists of various objects, as illustrated in Fig. 8.2. Qubit objects are representations of physical qubits. They store information regarding the physical parameters related to a qubit, which are collected during the calibration and characterization processes [40, 41]. This includes coherence times T_1 and T_2, as well as the parameters of Pulses and Pulsesequences required for the implementation of single-qubit native gates. QubitPair objects include data regarding the adjacent pairs of qubits on a chip, along with the associated two-qubit native gates. The chip's topology is derived from the available pairs and consumed by the transpiler discussed in Sec. 7.1.

Platform contains a collection of Instrument objects that include the low-level drivers necessary for operating the laboratory equipment. The abstract Instrument class consists of the methods required for interfacing Qibolab with the libraries supplied by the instrument's manufacturers. This ensures that the instrument can be integrated into a broader instrument setup, maintaining compatibility with all functionalities offered by the Qibo framework. The Controller class serves as a subclass for instruments equipped with arbitrary waveform generators, enabling them to play and acquire pulses. Qibolab offers pre-coded driver implementations for various commercial qubit control instruments, as detailed in appendix A.1.

Finally, the class Channel reflects the connection between Qubits and

`Instruments`. The `Port` object provides an interface for the control of `Instrument` parameters. The connection is critical for transmitting pulses from the instrument port that specifically targets the intended qubit. The system offers a qubit-centric interface for configuring instrument parameters, which is advantageous for calibration procedures.

To operate a real QPU, it is necessary to create a `Platform` that replicates the channel and instrument configuration of the laboratory, as demonstrated in Fig. 8.1. The procedure is described by these steps:

1. instantiate `Instrument` objects for all instruments in the lab setup;
2. create `Channel` objects for all connections between instruments and qubits, and map them to the corresponding instrument ports. Auxiliary instruments such as local oscillators can also be mapped to a `Channel`;
3. create a `Qubit` object for each qubit;
4. assign all applicable channels (readout, feedback, drive, flux, twpa) to each `Qubit`. Note that a qubit may not have all of these channels and a channel may be shared among different qubits.

Certain parameters involved in this procedure, including qubit-channel and channel-instrument connections, as well as instrument IP addresses, are static. Conversely, other parameters, such as those associated with the pulses implementing the native gates, vary dynamically throughout the qubit calibration process. It is essential to differentiate between these two categories and manage them independently within the code. Static parameters are generally hard-coded during the `Platform` generation process, whereas dynamic parameters are retrieved from external data sources. Additional information regarding the development of a custom `Platform` tailored for a specific laboratory configuration is available in the online documentation [42]. When the parameters of an existing platform are updated, such as through a calibration routine, the new parameters can be saved to disk thanks to specific serialization methods [43]. Parameters are transmitted to the designated devices thanks to their corresponding API, which is encapsulated by the `Platform` interface.

Running a program on the `Platform` involves multiple steps. Users have the capability to develop their programs with the `Qibolab` `Pulse` API or the `Qibo` `Circuit` API. The former is typically used for low-level applications, including qubit calibration, whereas the latter is essential for the execution of quantum algorithms.

8.4. Cross-platform benchmark

This section presents the results of a speed benchmark conducted using `Qibolab`. Several experiments were conducted on the various control

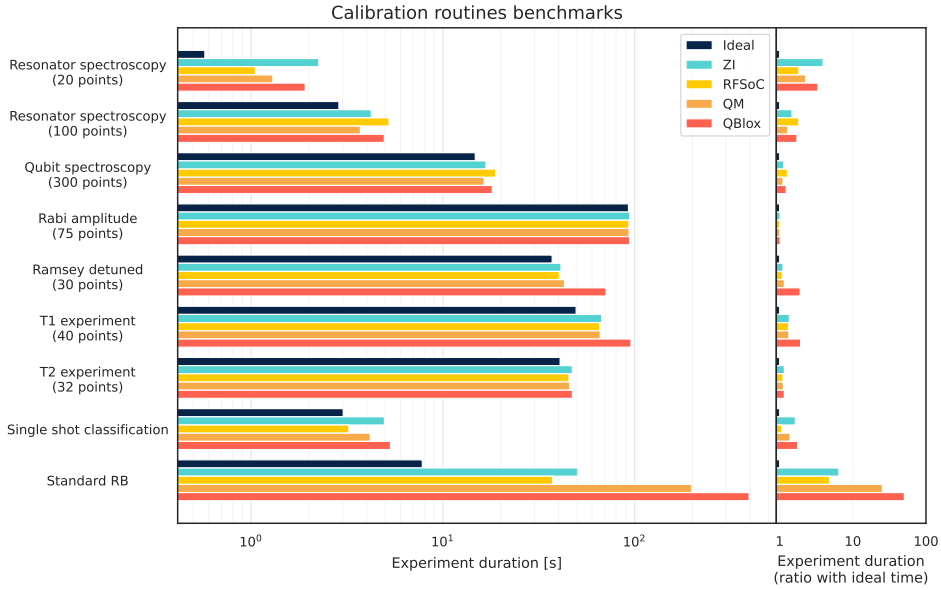


Figure 8.3.: Execution time of different qubit calibration routines on various electronics. On the left side we show the absolute times in seconds for each experiment. The ideal time (black bar) shows the minimum time the qubit needs to be affected in each experiment. On the right side we calculate the ratio between actual execution time and ideal time. Real-time sweepers are used, if supported by the control device, in all cases except the *Ramsey detuned* and *Standard RB* experiments. The benchmarked control devices are Zurich Instruments (ZI), the Radio Frequency System on Chip (RFSoC) operated through `Qibosoq`, Quantum Machines (QM) and Qblox. More details about these devices are collected in App. A.1.

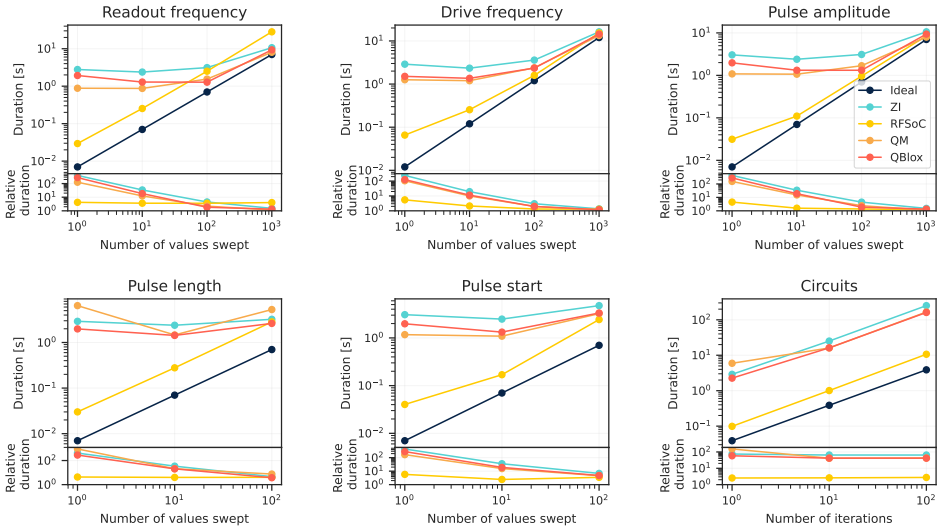


Figure 8.4.: Scaling of execution time as a function of the number of points in a sweep. Bottom plots show the ratio between real execution on different instruments and minimum ideal time. Real-time sweepers are used in all cases, except the last *Circuits* plot where we use the standard RB experiment to generate a given number of random circuits to execute.

devices that are currently supported by the drivers implemented in `Qibolab` as part of the benchmark.

With `Qibolab`, evaluating the effectiveness and performance of every control device is simple because all devices are accessible via the same interface. In addition to providing useful information on the various instruments' speeds, which can help researchers and developers in making well-informed choices, the results also show how well-supported these devices are within `Qibolab`.

The tests selected for this benchmark show the bare minimum of protocols needed to calibrate one qubit. Additionally, they provide an overview of the various modes of execution that `Qibolab` supports. Specifically, the *Single shot classification* experiment uses fixed pulse sequences, whereas the *Spectroscopies* experiment uses various sweeps over pulse parameters.

We compare the execution times of several qubit calibration procedures carried out with various circuits in Fig. 8.3, a brief explanation of these experiments can be found in App. A.3 . The black bar in this plot represents the ideal time required for each routine, which, is calculated with the formula:

$$\text{ideal} = n_{\text{shots}} \sum_i (T_{\text{sequence},i} + T_{\text{relaxation}}), \tag{8.1}$$

where $T_{\text{sequence},i}$ is the duration of the whole pulse sequence in the i-th point of the sweep, $T_{\text{relaxation}}$ the time we wait for the qubit to relax to its ground state between experiments, n_{shots} the number of shots in each experiment and the sum runs over all points in the sweep. The baseline for our benchmark is the ideal time, which indicates how long the qubit is actually used during an experiment. Because of overhead from compilations and transmission to the instruments, real executions, which are shown by a different color for each instrument setup, are longer than the ideal one. The overhead from the `Qibolab` backend, T_{qibo}, is insignificant in comparison to the control instruments, T_{inst}, as we can see after profiling the code. Therefore, we can approximate the real execution time as

$$\text{real} = T_{\text{qibo}} + T_{\text{inst}} + \text{ideal} \approx T_{\text{inst}} + \text{ideal} \tag{8.2}$$

When it comes to performing routines that need sweeps, there is a decisive aspect. That is, whether the sweeps operate in real-time on the host computer (software sweeper) or in the processors built into the control electronics. The latter method necessitates more communication steps between the host and control electronics and usually involves recompiling the applications several times, which adds a substantial overhead. This is demonstrated in the *Ramsey detuned* and *standard Randomized Benchmarking (RB)* experiments, where a considerable overhead over the ideal time results from the lack of real-time sweepers. In contrast to the other algorithms used here, randomized benchmarking experiments play numerous random sequences rather than sweeping

parameters, and it is anticipated that their performance will improve when sequence unrolling is applied.

Communication with the host computer is the second factor influencing performance. This typically consists of two steps: an instrument-side compilation step and the actual network (ethernet) communication. When real-time sweepers are not used, we find that RFSoC boards controlled by `Qibosoq` have an edge in this, especially from *Ramsey detuned* and *Single shot classification*. This benefit might result from our RFSoC configuration's simplicity: it only consists of one board, unlike other systems that are a part of clusters with multiple controllers. This point has to be confirmed by other research. The remaining electronics operate equally across all performed benchmarks, as anticipated.

We show how the execution time of various sweeps varies with the number of points used in the sweep in Fig. 8.4. As previously said, we observe that RFSoC is quicker for brief sweeps because of lower compilation and communication overhead; however, the difference becomes less noticeable as we approach 100 points. With the exception of *Circuits*, real-time sweepers are employed in every scenario shown in this graphic. We are now putting sequence unrolling techniques into practice, which will enable the execution of circuits in batches, lower communication overhead, and increase runtime.

8.5. Emulator

One of the remarkable achievements of `Qibolab` introduced in version 0.2 is the support for emulators that can effectively simulate quantum hardware. . This is an essential tool, particularly for `Qibocal`, as it can be adopted for testing the software itself (especially when access to the actual device is restricted or not possible). Additionally, it acts as a digital twin to specific quantum hardware, assisting in various functionalities such as predicting the outcomes of calibration experiments, performing coarse-grain calibrations, pulse-shaping, test-bedding, and beyond. For the latter, the emulator requires the device parameters that define the quantum hardware of interest as inputs to construct a precise model of the hardware.

To incorporate the emulator into the `Qibolab` ecosystem, we implemented a special controller class named `PulseSimulator`. The `PulseSimulator` is just called by the emulator, working primarily as an intermediary to translate and communicate objects between `Qibolab` and the chosen quantum dynamics simulation library (hereafter named the simulation engine) employed to numerically resolve the fundamental quantum dynamics of the device model for time-dependent control pulse sequences. It initializes the simulation engine using a device model defined by the device parameters and simulation settings in the runcard, extracts modulated signal waveforms from `Qibolab` pulse sequences, and transmits them to the simulation engine, which performs the dynam-

ics simulation. After completion, it converts the results produced by the simulation engine back into `Qibolab` (or `Qibo`) result objects.

Consequently, from the user's perspective, the emulator platform performs similarly to a quantum hardware platform, keeping them identical in features and functionalities. Therefore, it requires a platform folder and is initialized like any other device platform. Furthermore, the emulator produces as simulated outcomes a time sequence of state vectors (or density matrices when accounting for dissipation) of the fundamental quantum system, which are generated sequentially by the simulation engine as it resolves the dynamics delineated by the Lindblad master equation [1].

$$\dot{\rho}(t) = -i[H(t), \rho(t)] + \sum_k \frac{\gamma_k}{2}\left[2A_k\rho(t)A_k^\dagger - \rho(t)A_k^\dagger A_k - A_k^\dagger A_k\rho(t)\right]. \quad (8.3)$$

In the above, $\rho(t)$ is the density matrix of the system's quantum state at time t, $H(t)$ is its time-dependent Hamiltonian, and A_k are the operators through which the environment couples to the system with rate γ_k. As `Qibolab` currently only supports superconducting-qubits-based hardware, the device model used by the emulator is based on N capacitively coupled transmon qubits modelled as Duffing oscillators [5, 44] with number of energy levels predefined by the user in the `PulseSimulator` settings:

$$H(t) = H_{sys} + H_{drive}(t), \quad (8.4)$$

$$H_{sys} = \sum_{i=1}^{N}\left(\omega_i b_i^\dagger b_i + \frac{\alpha_i}{2} b_i^\dagger b_i (b_i^\dagger b_i - 1)\right) + \sum_{i\neq j} g_{ij}(b_i^\dagger b_j + b_i^\dagger b_j), \quad (8.5)$$

$$H_{drive}(t) = \sum_{i=1}^{N}\left[\Omega_{X,i}(t)\cos(\omega_{d,i}t) + \Omega_{Y,i}(t)\sin(\omega_{d,i}t)\right](b_i^\dagger + b_i). \quad (8.6)$$

Where, $\omega_i, \omega_{d,i}$ and α_i denote the resonant frequency, drive frequency and anharmonicity respectively for transmon i, $b_i(b_i^\dagger)$ its annihilation(creation) operator, and $\Omega_{X,i}(t), \Omega_{Y,i}(t)$ the drive amplitudes on its quadratures, while g_{ij} denotes the coupling strength between transmon i and j. Decoherence is incorporated using the Bloch-Redfield model [5], whereby each transmon is characterized by its longitudinal and transverse relaxation times $T_{i,1}$ and $T_{i,2}$ respectively,

$$A_{i,1} = \frac{1}{2}(\sigma_{i,X} + i\sigma_{i,Y}), \quad A_{i,2} = \sigma_{i,Z}, \quad \gamma_{i,1(2)} = \frac{2\pi}{T_{i,1(2)}}, \quad (8.7)$$

with $\sigma_{i,\mu=X,Y,Z}$ the Pauli matrices corresponding to transmon i and $A_{i,1}$, $A_{i,2}$ denote the A_k operators of Eq. 8.3 for transmon i. As an example, we

show in Fig. 8.5 the state overlap between the transmon qubit, modelled as a three-level system, with each of its energy modes as it interacts with the pulse for the X gate followed by the Hadamard gate. This can be easily extended to include other quantum computing technologies when they will be available on `Qibolab`. The first supported simulation engine is based on `QuTiP` [45], with plans to incorporate `JAX` [46] support for GPU acceleration, as well as tensor-network based quantum dynamics simulation libraries to speed up the simulation of larger system sizes with a lower memory footprint but with a small accuracy cost.

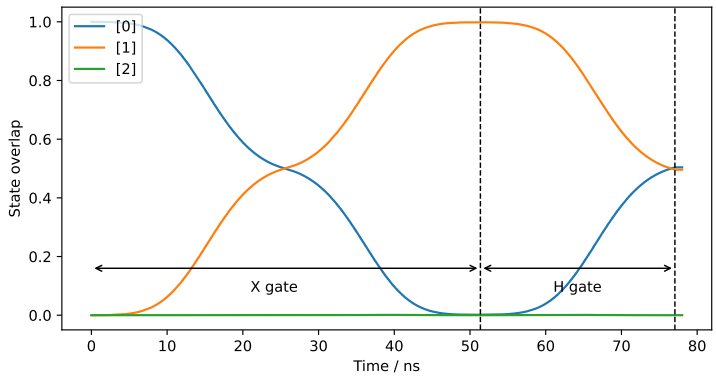


Figure 8.5.: State overlap between the simulated qubit modelled as a three-level system with each of its energy modes as it evolves under a control pulse sequence for an X gate followed by a Hadamard (H) gate.

At the time of writing, resonators are excluded from the simulation model, hence the only supported acquisition modes are discrimination and integration, with the latter currently implemented as a projection onto the in-phase pulse component. Despite this constraint, the existing emulator is capable of executing the majority of `Qibocal` protocols for the purposes of simulating calibration and device characterization. This section of the library remains under development, and we intend to enhance it with more features. Enhancing the complexity of the simulated quantum system by incorporating flux-tunable qubits, resonators, and couplers is a component of our strategic plan.

9. Qibocal

9.1. Software design

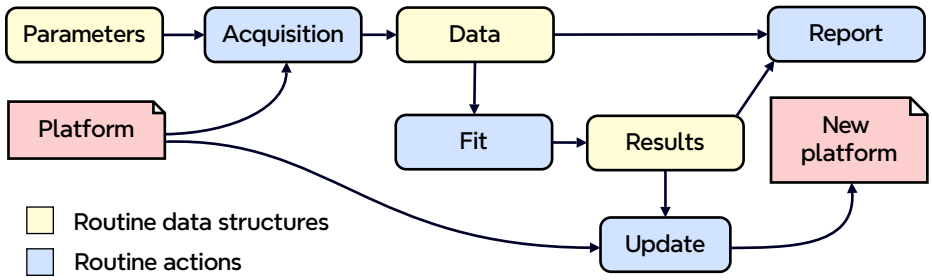


Figure 9.1.: Dependence scheme for a `Qibocal` `Routine`.

`Qibocal` is the software element of the Qibo architecture that is in charge with the characterization and calibration of a quantum processor managed by Qibolab.

Fundamentally, `Qibocal` includes multiple *protocols* that correspond to individual experiments designed to extract or correct physical parameters crucial to the characterisation and calibration of superconducting quantum devices. Additionally, the library offers all essential functionalities to efficiently execute protocols, obtain, and share results. Qibocal manages the storage of all parameters and maintains an updated configuration of quantum processing units (QPU).

Among the operations required for a fully working quantum computer, three distinct types of experiments can be identified. Initially, characterization experiments are performed to extract individual parameters of the system's model. Secondly, calibration tests which mostly involve fine-tuning to maximize specific metrics like the readout fidelity or the pulse fidelity. Ultimately, we may consider more generic experiments that cannot be entirely categorized as either calibration or characterisation studies, such as validation experiments. Apart from this final option, we will refer to any experiment of the two preceding categories as a *protocol* for the sake of simplicity.

A protocol in `Qibocal` is represented as a `Routine` class, designed to act as a representation of a general experiment executable on a quantum computer. Such experiments typically encompass two phases: data acquisition and data processing. In this context, *data acquisition* denotes a general measurement on a quantum device, including both

fundamental experiments involving simple pulse sequences or circuit executions and more sophisticated ones. Data processing refers to any calculation conducted after measurement, meaning when access to the QPU is no longer required. It is often challenging to distinguish between acquisition and processing in a complex experiment, as the latter may influence the former. It is possible to think of an advanced experiment as being made up of smaller components, and protocols are meant to be the atomic actions.

Furthermore, several optional steps are available: when a decent calibration is reached, they can generate an updated platform configuration based on the outcomes achieved. Additionally, it is possible to plot both the data and the post-processing analysis via tables and graphs. All these procedures are depicted in Fig. 9.1.

We discuss briefly all the classes which are involved in the definition of a `Routine` object.

The external input to a `Routine` is denoted by its `Parameters`, which are consumed during the data acquisition phase. Examples of such parameters may include sweeping ranges for spectroscopy experiments, the selection of qubits on which the protocol will be executed, and non-protocol-specific configuration parameters such as the number of shots to be executed and the relaxation time between different measurements.

The unprocessed data generated by the acquisition are stored in a `Data` class. By default, the collected data will be kept in binary files for rapid loading and unloading, mostly in a NumPy-specific format, while supplementary parameters are stored as JSON files. Nonetheless, Qibocal gives the freedom to save and retrieve data in any generic Python-compatible format, assuming the user supplies the appropriate techniques for loading and dumping.

The information derived from the obtained `Data` through post-processing analysis is subsequently organized in the `Results`. These items are designed to be shallow, incorporating only a limited number of parameters that contain the results of the specific protocol. The results are utilized to upgrade quantum computer configurations with the new settings.

9.2. Command line execution

`Qibocal` provides a simplified way to launch protocols through a command line interface (CLI), which is summarized in Fig. 9.2. All protocol-related information before execution, which basically consists of the `Parameters` and the QPU configurations, are serialized into a `YAML` file which we call *experiment runcard*. The `Runcard` is then deserialized by an `Executor` which takes care of instantiating a task which is associated to a specific `Routine`. After a task has been dispatched, the `Executor` stores all data obtained from acquisition and post-processing. The `Executor` also takes care of updating the QPU.

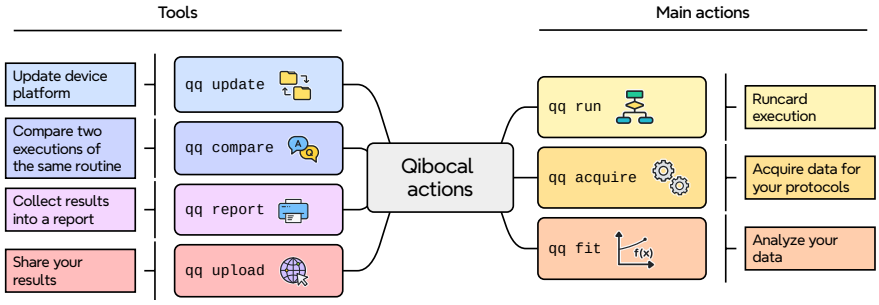


Figure 9.2.: A diagram summarizing the Qibocal CLI, including a brief description of each command.

Through the CLI it is possible to run experiments with both acquisition and data processing using `qq run`, or the two steps can be performed separately using `qq acquire` or `qq fit` respectively.

Inside an *experiment runcard* it is possible to include more than one experiment, which will be executed sequentially. In this case, the QPU is updated immediately after each protocol, which means that the subsequent experiments will be performed with a `Platform` that contains the parameters calibrated by the previous experiment. This approach is intended to be employed mostly for monitoring application as well as short recalibration programs. We are going to address how to handle non-sequential schemes in Sect. 9.3.

9.3. A calibration program

Ad hoc measurements often require many protocols and dependencies that go beyond sequential execution. This happens usually during a full-chip calibration, especially when the number of qubits and connected qubit pairs is high.

Graphs have frequently been used to model calibration workflows [47], with directed acyclic graphs (DAGs) emerging as a particularly effective representation. DAGs naturally encode dependency relationships, allowing for clear and potentially optimized execution strategies (e.g., through topological sorting).

However, DAGs can only represent workflows where the full execution path is known in advance. This constraint excludes cyclic dependencies, commonly approximated by unfolding them into successive layers of increasing precision [47], as well as any form of runtime conditional logic, such as loops.

It is also worth noting that, in typical scenarios, the sequence of

operations during the initial calibration phase is well-established and relatively uniform across different chips. This consistency reinforces the idea of representing the calibration process as an executed program.

In contrast, later refinement stages often involve more intricate parameter interdependencies, making it harder to clearly separate operations. Strategies at this stage tend to be more heterogeneous and less predictable. Enabling runtime execution therefore becomes crucial to accommodate this increased complexity and flexibility.

To this end, `Qibocal` is designed to support a flexible and expressive execution model, embedding calibration logic directly within a high-level programming language, offering a natural interface for both standard workflows and adaptive, runtime-driven procedures.

This approach maps each atomic operation (i.e. protocol) to a callable function, aligning with the computational model of procedural programming languages. The precise behavior of this function may depend on the internal state of the calibration executor, which orchestrates the overall process.

As illustrated in Fig. 9.1, the protocol execution follows a defined structure; however, individual stages are optional and can be deferred. For example, report generation is typically performed at a later stage, while acquisition can be skipped entirely if the script is applied to pre-existing data.

Currently, this model is implemented in Python, where the executor is instantiated as an object, and protocols are exposed as its methods. These methods accept input parameters and return post-processed results as outputs.

Beyond protocol execution, the executor provides additional operations that collectively define an embedded domain-specific language (eDSL) available to `Qibocal` script users. Examples include device connection and disconnection procedures, as well as result serialization.

Although Python is currently used due to its role as the implementation language of `Qibocal`, protocol exposure is not limited to it. In principle, protocols can be made accessible from other languages by binding to the existing Python library, e.g., through lightweight wrappers. However, such bindings have not yet been implemented or tested, and their development is deferred to the broader Qibo ecosystem focused on cross-language interoperability.

In addition to enhancing flexibility and simplifying the specification process, leveraging the expressiveness of programming languages enables simple interface with external technologies. Intermediate results from protocols can be processed using external libraries during the calibration cycle and can be directly linked to auxiliary systems, such as data providers, or monitoring systems as in Sect. B.1.4.

Protocols remain uniquely characterized by their ability to define and tune the acquisition experiment, that is, the precise sequence and timing of pulses to be executed. This specification is encapsulated within the protocol and represents a fundamental unit of computation. Each

protocol is typically paired with a corresponding post-processing routine, which is often kept together for convenience, though it can be replaced or extended as needed. The same applies to associated update steps and report generation.

While the flexibility discussed above reduces the need for a large number of fixed protocols, since post-processing can be decoupled and handled at a higher level, it also enables extensibility. New protocols, beyond those provided by `Qibocal`, can be easily integrated. To be used at runtime, these protocols simply need to be registered and included in the executor object.

An example of a typical `Qibocal` script for the execution of a calibration protocol would look like:

```
from qibocal import Executor
from qibocal.cli.report import report

# Add here platform and path
platform = "my_platform"
path = "my_path" # Path to dump all the data

# The Executor takes care of the deployment on hardware of the experiments
    .
with Executor.open(
    "myexec",
    path=path,
    platform=platform,
    update=True, # Update the platform after each experiment execution
    force=True, # Force the Executor to ovewrite the data folder
) as e:
    # Define the calibration experiments, a single shot one
    # in this example, with their parameters
    ssc = e.single_shot_classification(nshots=1000)
    print("\nfidelities:\n", ssc.results.fidelity, "\n")
```

Some examples on the use of `Qibocal` scripts are collected in Sec. B.1.1 B.1.2 B.1.3.

9.4. Tools

`Qibocal` offers a collection of tools that are intended to make the implementation of calibration procedures more efficient. A web page with an in-depth summary of all executed protocols is produced when a program is run via the CLI.

The `report` function, which is a component of the `Routine` interface depicted in Fig. 9.1, allows for complete customization of the content. The final report automatically compiles the plots and tables generated by each protocol. Users can also directly embed HTML strings for more extensive customization, giving them total control over the results for particular experiments.

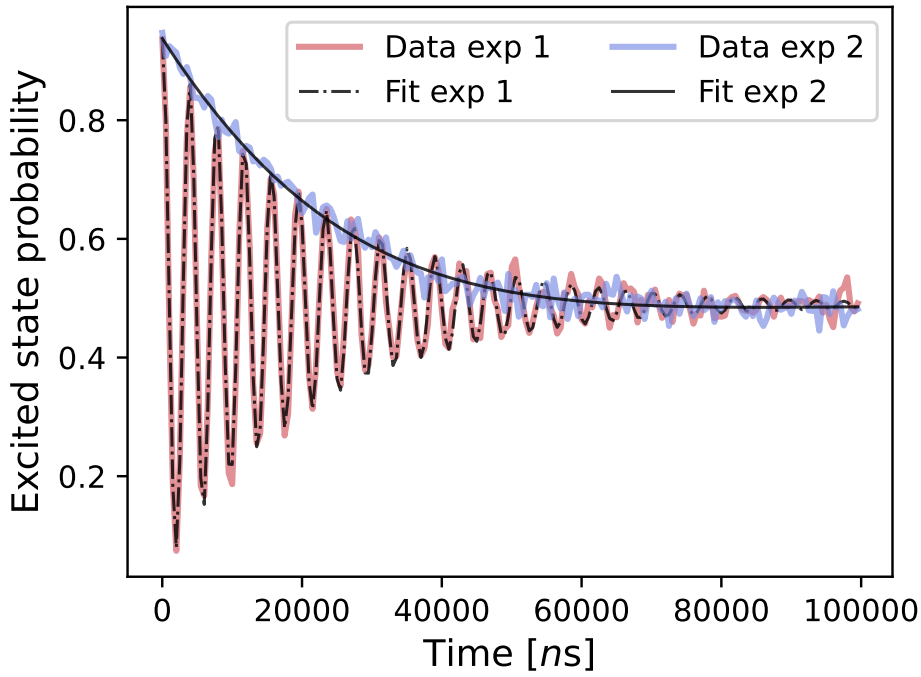


Figure 9.3.: Comparison between two Ramsey experiments executed using the command `qq compare` in `Qibocal`. In blue we present a first run of the experiment. The presence of oscillations is an indication of the fact that the frequency of the drive pulse is not aligned with the qubit frequency. In red we run the same experiment after correcting the qubit frequency.

The CLI command `qq upload` can be used to upload reports to a dedicated server for easy sharing. We find that the current lightweight solution is adequate for our needs, even though a database-backed system would be the more traditional option. In the future, we also intend to allow uploads straight from custom protocols.

The CLI command `qq compare` creates a combined `HTML` report by combining plots and tables from two separate runs to make comparison easier. This feature is especially helpful for detecting modifications to the experimental setup, as shown in Fig. 9.3.

Finally, the `qq update` command is used to replace the current platform, overwriting it with the one generated by `Qibocal` and dumped in the report folder at the end of the calibration experiment.

9.4.1. Automatic recalibration

This set of protocols can serve as a recalibration tool. Qibocal can pass calibrated parameters from one routine to the next, either sequentially or through more complex connections. The calibration program API (Sect. 9.3) provides the flexibility to design custom calibration schemes and to alter the default pipeline based on results from earlier experiments. Running protocols directly from Python also allows the implementation of ad-hoc stopping conditions tied to specific thresholds. Verification tools, including parameter-reliability tests, can transform this into a fully automated recalibration process.

Part III.

Applications

10. Qubit calibration with `Qibocal`

10.1. Introduction

Before executing any algorithm on a quantum platform, a tune-up procedure is required to precisely determine the optimal platform parameters. The exact name of this procedure depends on the goals and the initial state of the device.

When a superconducting qubit is placed in the cryostat for the first time, the initial task is to directly or indirectly measure the parameters that define the Hamiltonian of the system, such as the charging energy E_C or the qubit–resonator coupling constants. Although design and fabrication provide approximate values, these often differ significantly from the actual ones. This process typically involves extensive scans and numerous experiments, and is referred to as **characterization**.

Once the chip has been characterized, its parameters may drift over time, or the previous characterization may lack sufficient precision. In such cases, fine-tuning experiments, such as Ramsey sequences, are performed to improve the performance of the chip. These procedures operate under the assumption that the true parameters are close to the previously measured ones, and are referred to as **calibration**.

`Qibocal` provides protocols for both characterization and calibration. While not strictly necessary, having prior knowledge of the qubit and resonator frequencies is advantageous, as it avoids the need for wide frequency sweeps.

In this chapter, we describe how to calibrate a superconducting platform with flux-tunable qubits and fixed qubit-qubit couplings, with experiment results acquired and post-processed by `Qibocal`. The plots, tables and results shown are obtained from the calibration of a pair of interacting superconducting qubits operated with `Quantum Machines` electronics. The measured parameters are summarized in Table 10.1.

10.2. Single qubit calibration

The first step in achieving full control over a qubit is calibrating its single-qubit gates. In practice, R_X gates combined with R_Z rotations can implement any arbitrary single-qubit rotation [5]. While R_Z rotations can be realized in several ways [1], the most efficient approach is a virtual

	Q_1	Q_2
Res. freq. (GHz)	7.12	7.47
Qubit freq. (GHz)	5.0007	5.685
Anharmonicity (MHz)	108	148
Single qubit error (%)	0.086 ± 0.001	0.108 ± 0.001
Readout fidelity	0.918	0.952
T_1 (μs)	21.8 ± 0.1	25.7 ± 0.2
T_2^* (μs)	27.4 ± 0.4	21.1 ± 0.4
T_{2e} (μs)	24.1 ± 0.3	24.3 ± 0.4

Table 10.1.: Parameters of the qubits Q_1 and Q_2 used to produce the shown calibration results.

implementation [48], achieved by shifting the phase of subsequent pulses during circuit compilation. Consequently, calibration efforts focus on R_X rotations and measurement gates.

10.2.1. Resonator

The first step is to determine the resonator's bare frequency ω_R^{bare} through high-power one-tone spectroscopy (Fig. 10.1). This measurement requires no prior information about the device, pulse parameters, or amplifier configuration. Operating at high power enhances the signal-to-noise ratio (SNR), thereby reducing the influence of background noise.

For two-dimensional notch-type resonators, we apply the fitting procedure described in Ref. [49], which provides not only the resonant frequency but also the quality factors and impedance mismatch. This is achieved by fitting the measured complex transmission to the model

$$S_{21}(f) = ae^{i\alpha}e^{-2\pi i f\tau}\left[1 - \frac{(Q_l/|Q_c|)e^{i\phi}}{1 + 2iQ_l(f/f_r - 1)}\right] \tag{10.1}$$

where a denotes the signal amplitude far from resonance, α accounts for possible phase shifts, and τ represents the cable delay determined by cable length and the finite propagation speed of light. The term ϕ accounts for impedance mismatches or reflections in the transmission line (Fano interferences), and f_r is the resonator frequency. The parameter Q_l is the *loaded quality factor*, which characterizes the total energy loss in the resonator and Q_c is the *coupling quality factor* associated with the coupling between the resonator and the environment.

In addition, `Qibocal` provides a Lorentzian fit of $|S_{21}(f)|$ to directly extract the frequency peak.

When performing this measurement, two parameters must be considered: the *reset-time* (sometimes referred to as the repetition duration) and the *number of shots*. The number of shots specifies how many times the same experiment is repeated at a given frequency, while the

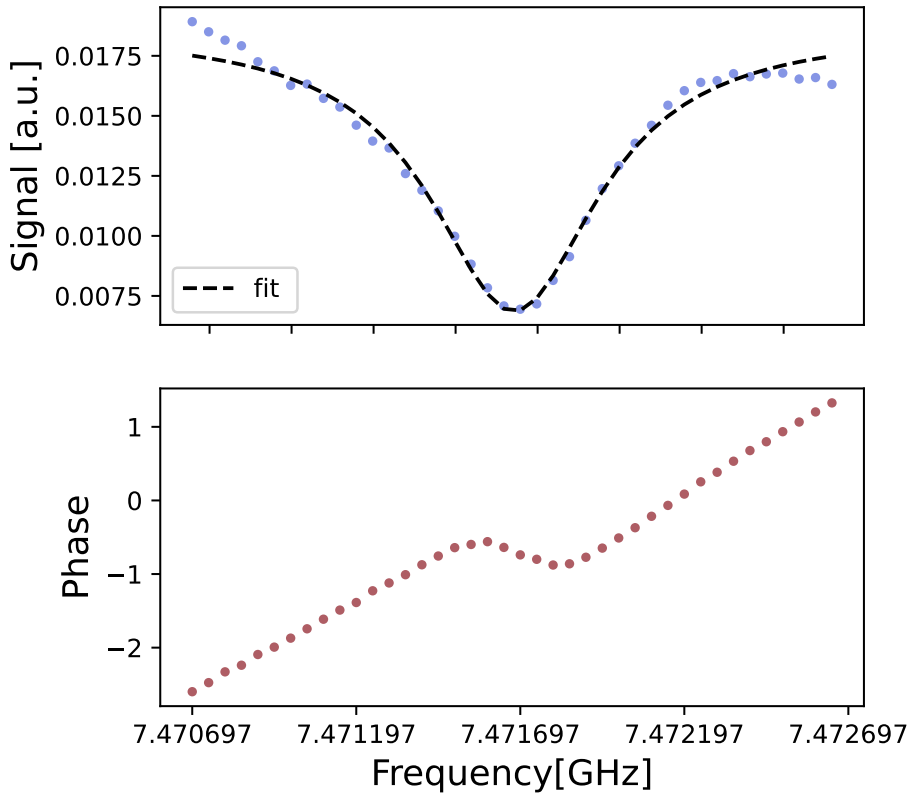


Figure 10.1.: Resonator spectroscopy results. The signal with its Lorentzian fit and the phase are represented.

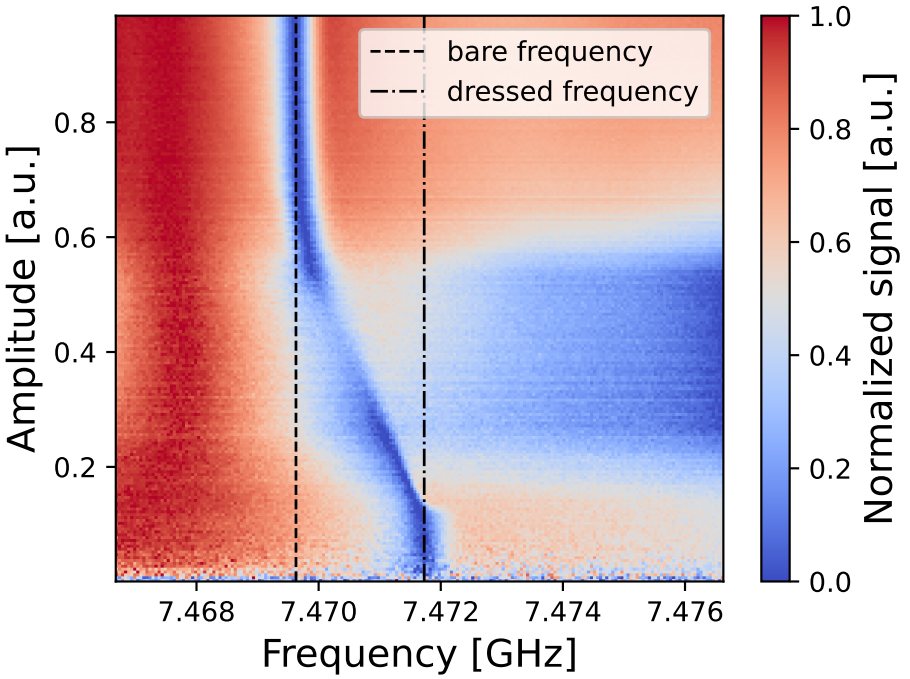


Figure 10.2.: Results of a punch-out experiment showing the shift of the resonator frequency at different amplitude values.

reset-time is the interval between repetitions. Increasing the number of shots improves the SNR through averaging, but also extends the overall measurement time. For this particular experiment, the *reset-time* can be set to zero, since the qubit is never excited and relaxation processes are therefore irrelevant.

Calibration of the resonator frequency continues with the determination of the dressed resonator frequency ω_R^{dress} using the *punch-out* experiment (Fig. 10.2). In this procedure, the spectroscopy measurement is repeated at varying input powers. The data typically exhibit two regimes: at high power, resonance peaks occur at ω_R^{bare}, while after a transition region, a low-power regime emerges from which the dressed frequency can be extracted. The frequency difference between the two regimes corresponds to the Lamb shift,

$$\omega_R^{\text{bare}} - \omega_R^{\text{dress}} = \frac{g^2}{\Delta}. \tag{10.2}$$

This relation implies that once both the dressed and bare frequencies are measured, and the coupling constant g between the qubit and resonator is known (e.g., from design specifications), the qubit frequency

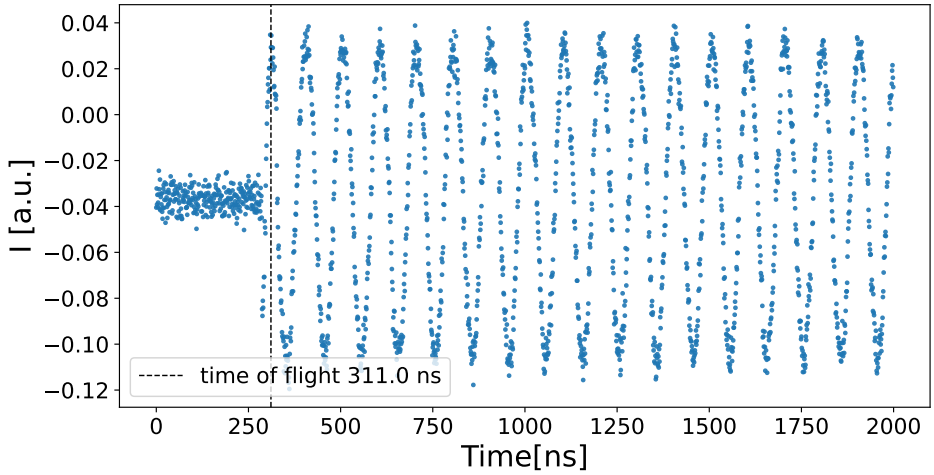


Figure 10.3.: Measurement of the time of flight, clearly showing the exact moment the signal reaches the acquisition port.

can be determined. Moreover, the experiment serves as a diagnostic: the absence of a frequency transition indicates that the resonator is not coupled to the qubit.

The *punch-out* experiment provides both the frequency and the readout pulse amplitude that maximize the SNR in the low-power regime. A more precise estimate of the dressed frequency can then be obtained by repeating the one-tone spectroscopy at this optimal amplitude, restricted to a narrower frequency window.

An additional measurement that enhances readout performance and yields further information about the resonator is the *time-of-flight* experiment (Fig. 10.3), that in principle can be executed before the resonator spectroscopy. In this procedure, the propagation time of a readout pulse is measured as it travels to the qubit chip and back, where it is recorded by the control instrument ADCs.

Once the instrument begins detecting oscillations of the readout pulse, rather than background noise, the cable delay can be determined. If the acquisition window exceeds the sum of this delay and the pulse duration, the characteristic exponential decay due to photon losses can be observed, enabling an estimate of the loss rate κ. However, κ can alternatively be extracted from resonator spectroscopy, where it is directly related to the Lorentzian linewidth of the resonance.

10.3. Qubit

10.3.1. Frequency

Analogous to the resonator calibration, the first step in calibrating the qubit is to determine its transition frequency. This is done using two-tone spectroscopy, where a drive pulse is applied to the qubit drive line, followed by a measurement pulse on the readout line. By sweeping the drive frequency ω_d, the qubit frequency ω_q is identified at resonance (Fig. 10.4). When $\omega_d \approx \omega_q$, the qubit is excited, and through dispersive coupling, the resonator frequency shifts.

The outcome of this measurement depends strongly on the drive amplitude. Ideally, the amplitude should be sufficient to excite the qubit to the first excited state without introducing power broadening. If the amplitude is too low, the qubit remains unexcited; if too high, the resonance linewidth broadens due to power broadening [50]. Since the correct amplitude is not yet known at this calibration stage, the drive pulse duration is chosen to be long compared to the qubit coherence times. This ensures that, just prior to measurement, the qubit state can be assumed to be maximally mixed, $\rho \sim I$.

To determine the optimal drive amplitude, `Qibocal` provides a variant of qubit spectroscopy in which both the drive frequency and amplitude are swept. This approach not only identifies the proper excitation amplitude but also reveals higher-order transitions. In particular, the $0 \rightarrow 2$ transition, which is a two-photon process, is expected to produce a resonance peak at $\omega_{02}/2$.

10.3.2. Sweetspot

As described in Sec. 3.3, the SQUID qubit frequency depends on the applied external flux. By measuring this dependence, it is possible to indirectly extract important parameters such as the relative junction asymmetry d or the Josephson energy, as can be seen from Eq. 3.23. Moreover, this procedure allows us to identify the *sweet spot*, which can be chosen as the qubit's operational point.

The experiment typically consists of qubit spectroscopy while sweeping the bias current applied to the flux line, the results are shown in Fig. 10.5. Although probing the qubit frequency directly is the most common approach, it is also possible to perform a resonator spectroscopy, whose frequency follows the relation

$$f_r(\Phi) = f_r^{\text{bare}} + g_0^2 \frac{\sqrt[4]{d^2 + (1-d^2)\cos^2\left(\pi \frac{\Phi}{\Phi_0}\right)}}{f_r^{\text{bare}} - f_q(\Phi)}. \tag{10.3}$$

In this way, the sweet spot can be identified from the resonator response before searching for the qubit frequency.

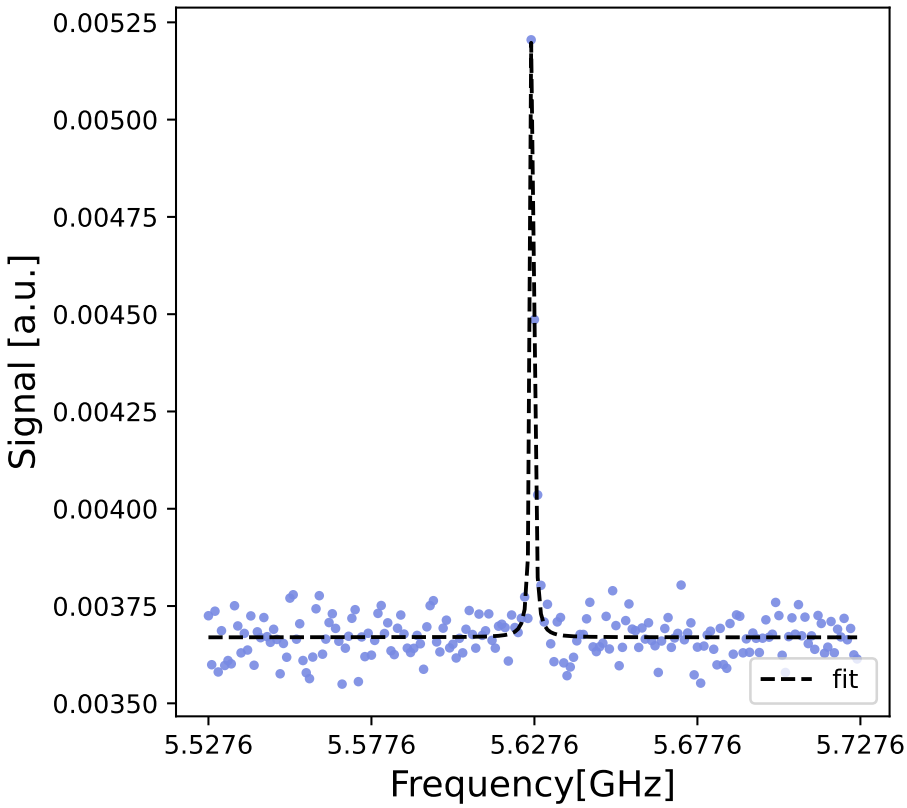


Figure 10.4.: Qubit spectroscopy experiment in which the qubit frequency is identified from the resonant peak.

Choosing the sweet spot (or alternatively the anti-sweet spot) is a reasonable choice at this level of calibration. However, it must be noted that the optimal operational point may differ due to effects such as crosstalk between control lines or the presence of defects, e.g. parasitic two-level systems (TLS).

10.3.3. Amplitude

The next goal is to determine the proper amplitude for implementing the $RX(\pi)$ pulse, commonly referred to as the π gate. Since what ultimately matters is the integral of the pulse envelope, one can either fix the amplitude and determine the corresponding duration, or vice versa. `Qibocal` supports both approaches; however, we usually prefer fixing the duration (here 40 ns) to ensure that decoherence effects remain negligible.

The amplitude is determined through a Rabi experiment (Fig. 10.6),

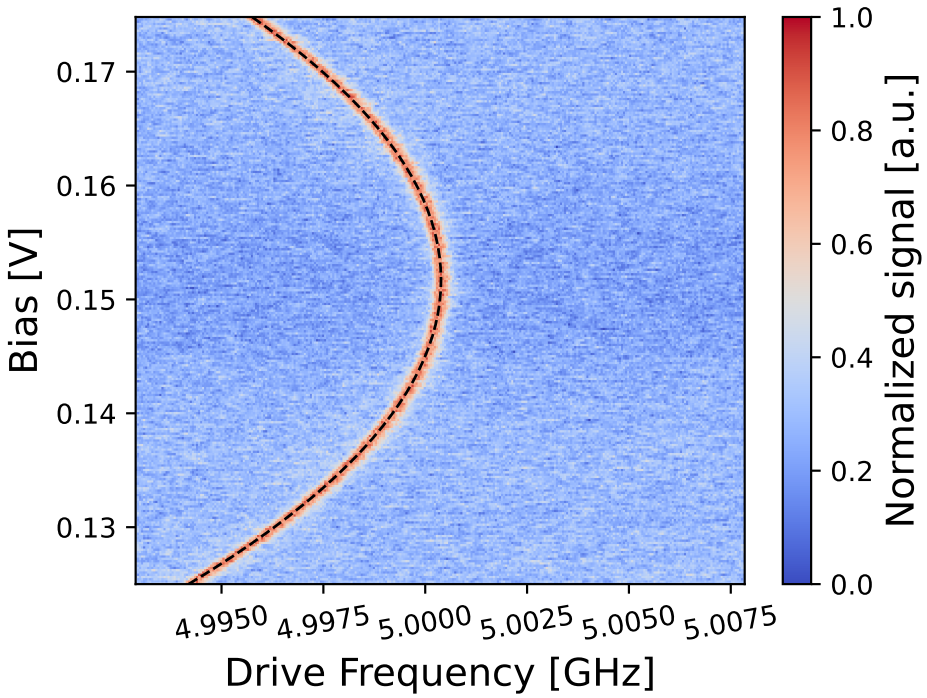


Figure 10.5.: Measurement of the qubit frequency (red regions) as a function of the flux amplitudes (bias). In this plot, it is possible to identify the sweetspot at $\sim 0.15V$.

in which the qubit is probed with a drive pulse (Gaussian in the initial calibration) at its transition frequency, followed by a measurement. This pulse sequence is repeated multiple times while varying the pulse amplitude. Starting from the ground state, the qubit's probability of occupying the first excited state oscillates sinusoidally as the drive amplitude is swept,

$$p_e(t) = \frac{1}{2}\left(1 - \cos\left(\Omega_R t\right)\right). \tag{10.4}$$

where Ω_R represents the Rabi frequency, that is proportional to the pulse amplitude.

Since many routines and protocols in quantum computing rely on $R_x(\pi/2)$ rotations, `Qibocal` also implements an alternative version of the Rabi experiment. This version can be used to tune the amplitude (or duration) of the drive pulse to excite the qubit from the ground state to the state $\frac{|0\rangle - i|1\rangle}{\sqrt{2}}$.

Calibrating an $R_x(\pi/2)$ rotation as a native gate allows us to avoid errors that could arise from assuming that the $R_x(\pi/2)$ amplitude (or

duration) is exactly half of that of the $R_x(\pi)$ gate. Such an assumption requires a perfectly linear qubit response to the drive pulse, which is often not satisfied due to nonlinearities in the qubit or imperfections in pulse shaping.

In this case, the pulse sequence is identical to the standard Rabi experiment, except that instead of a single $R_x(\pi)$ pulse, we apply two consecutive $R_x(\pi/2)$ pulses.

During the first execution of a Rabi experiment, the classification model for the $|0\rangle$ and $|1\rangle$ states has not yet been trained. Therefore, the data can be fitted using only the magnitude of the readout signal, $|S_{21}(f)|$, or the I, Q components.

10.3.4. State classification

In order to operate with probabilities in subsequent calibration experiments, we first need to train a model for qubit state classification. This is done using the *single-shot* experiment.

In this experiment, the qubit is repeatedly prepared in the states $|0\rangle$ and $|1\rangle$. For each shot, the I and Q values are recorded. Theoretically, taking noise into account, each state should produce a Gaussian blob in the IQ plane [5].

The classification method identifies the axis passing through the centroids of the two clusters in the training set and determines the optimal threshold that maximizes the difference between the cumulative distributions of the projections along this axis. To classify a new point, its projection along the axis is compared to the threshold. This method is simple and highly interpretable.

The line is characterized by the **angle** θ and the **threshold** coordinates. The angle θ (in radians) is formed by the line connecting the centers of the two Gaussian blobs and the Q-axis. After rotating the data by $-\theta$ and translating such that the origin coincides with the ground state Gaussian center, the threshold is defined as the distance along the x-axis that maximizes the classification fidelity [51].

$$\mathcal{F} = 1 - \frac{1}{2}(P(0,1) + P(1,0)) \tag{10.5}$$

where $P(i,j)$ is the probability of measuring the qubit in state i but prepeared in state j. With this routine it is possible to train different classifiers, plot their ROC curves and evaluate the Area Under the Curves.

10.3.5. Drive fine tunings

At this stage of the calibration, we should have a calibrated π and measurement pulses. However, benchmarking the calibration often reveals suboptimal metric values, indicating that there is still room for improvement.

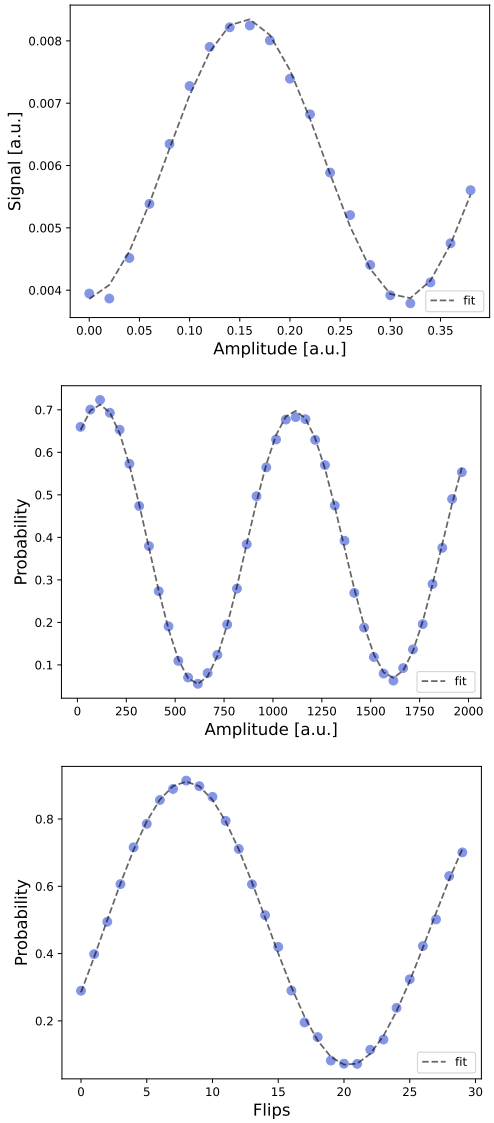


Figure 10.6.: **Top**: Rabi experiment, obtained by sweeping the qubit-drive amplitude and revealing classical Rabi oscillations; this step precedes state classification, hence the signal on the y-axis. **Middle**: Ramsey experiment used to fine-tune the qubit frequency. **Bottom**: Flipping experiment correcting small under- and over-rotations.

The qubit drive pulse frequency can be fine-tuned using a Ramsey experiment, since the qubit frequency obtained from spectroscopy carries an uncertainty related to the spectral linewidth.

In this protocol, two short $\pi/2$ pulses are applied, separated by a waiting time τ. The first $\pi/2$ pulse projects the qubit from $|0\rangle$ onto the xy plane of the Bloch sphere. During the interval τ, the qubit undergoes decoherence around the z axis. After the second $\pi/2$ pulse, a measurement is performed. By varying the waiting time τ, one observes an exponential decay in the probability of measuring $|1\rangle$, providing an estimate of T_2^*.

If there is a detuning between the qubit frequency and the drive pulse frequency, the measured probability will exhibit a sinusoidal modulation. This oscillation can be used to extract a correction to the qubit frequency:

$$p_e = \frac{1}{2} + \frac{1}{2} e^{-\tau/T_2} \cos\left(\Delta\omega \pm \frac{\pi}{2}\right) \tag{10.6}$$

Here, $\Delta\omega$ denotes the detuning between the qubit and the drive pulse frequencies. To reliably determine the qubit frequency correction, the drive pulse is often intentionally detuned to induce oscillations.

To fine-tune the qubit drive amplitude, an error amplification sequence experiment (flipping experiment in `Qibocal`) can be performed. In this protocol, an initial $R_x(\pi/2)$ rotation is applied, followed by N flips (each consisting of two $R_x(\pi)$ rotations), and the qubit state is measured. The initial $R_x(\pi/2)$ pulse is necessary to distinguish between under-rotations and over-rotations of the $R_x(\pi)$ pulse. Without it, the difference between the two cases appears only as a global phase, leaving the measurement probabilities unchanged. With the $R_x(\pi/2)$ pulse, under-rotations leave the state closer to $|0\rangle$ after the first flip, whereas over-rotations move the state closer to $|1\rangle$.

By fitting the measured data with a sinusoidal function, the amplitude correction ΔA can be extracted, allowing refinement of the π-pulse amplitude.

A similar version of the flipping protocol can also be used to calibrate the drive amplitude for $R_x(\pi/2)$ rotations, where each $R_x(\pi)$ rotation is replaced by two consecutive $R_x(\pi/2)$ pulses.

Whenever we operate with a transmon qubit, we must remember that it is only approximately a two-level system. In reality, it has multiple energy levels, and operations on the qubit can unintentionally excite higher levels, leading to **leakage errors**.

For example, consider applying a Gaussian pulse $\Omega(t) = \Omega e^{-t^2/2\sigma^2}$, if the pulse duration is sufficiently short, its Fourier transform can have a frequency bandwidth larger than the qubit anharmonicity (typically around 200–300 MHz), resulting in excitation of the $|2\rangle$ level.

Additionally, one must account for phase errors caused by the AC Stark shift of the $|1\rangle - |2\rangle$ transition. This interaction effectively repels the two levels, shifting the ω_{01} frequency.

To mitigate both leakage and phase errors, DRAG pulses (Derivative Reduction by Adiabatic Gate) are employed. The idea is to add a quadrature component to the pulse that is proportional to the time derivative of the in-phase component. For a pulse with in-phase component Ω_x, the quadrature component Ω_y is defined as

$$\Omega_y(t) = \beta \frac{d\Omega_x}{dt}, \tag{10.7}$$

where β is a scaling parameter. This parameter β can be extracted by playing the pulse sequence composed of N repetitions of $R_X(\pi) - R_X(-\pi)$ pulses for different values of β as shown in [52]. The post-processing consists of measuring the probability of $|0\rangle$ for every β and fitting the curve with a cosine. The correct β value is the one which maximizes the curve.

10.3.6. Readout fine tuning

We can further improve the calibration of the readout pulse by adjusting its parameters. Theory predicts [51] that increasing the duration and amplitude of the readout pulse improves the signal-to-noise ratio (SNR). However, this also increases the probability of qubit state transitions. This effect is evident when we plots the two blobs in the IQ-plane: instead of observing two well-separated Gaussian blobs, the distributions become elongated into ellipses. Therefore, optimizing the readout requires balancing these two competing effects. It is also clear that readout fidelity alone is not a sufficient metric; the shape of the blobs and the QND nature (quantum non-demolition property) of the measurement must also be considered.

Similarly, shifting the readout frequency can enhance the readout quality, as demonstrated by the *dispersive shift experiment*. In this protocol, resonator spectroscopy is performed twice: once with the qubit prepared in $|0\rangle$ and once in $|1\rangle$. The resonator frequency shifts depending on the qubit state, and the magnitude of the shift is related to $\chi = -\frac{g^2}{\Delta}$,

$$\omega_1^r - \omega_0^r = 2\chi. \tag{10.8}$$

From this protocol, we can extract both the qubit–resonator coupling constant and, for each readout frequency, the separation between the $|0\rangle$ and $|1\rangle$ clusters. The optimal readout frequency is the one that maximizes this separation. Theoretically, this occurs at the midpoint between the two resonator frequencies, where $|S_{21}|$ is the same but the phase difference is maximized.

Furthermore, examining the trajectories of the readout signals for the excited and ground states, we observe that at the beginning and the end of the signal, the two traces overlap closely, containing little discriminative information. To account for this, one can apply a weighting envelope that emphasizes the time windows carrying the most useful

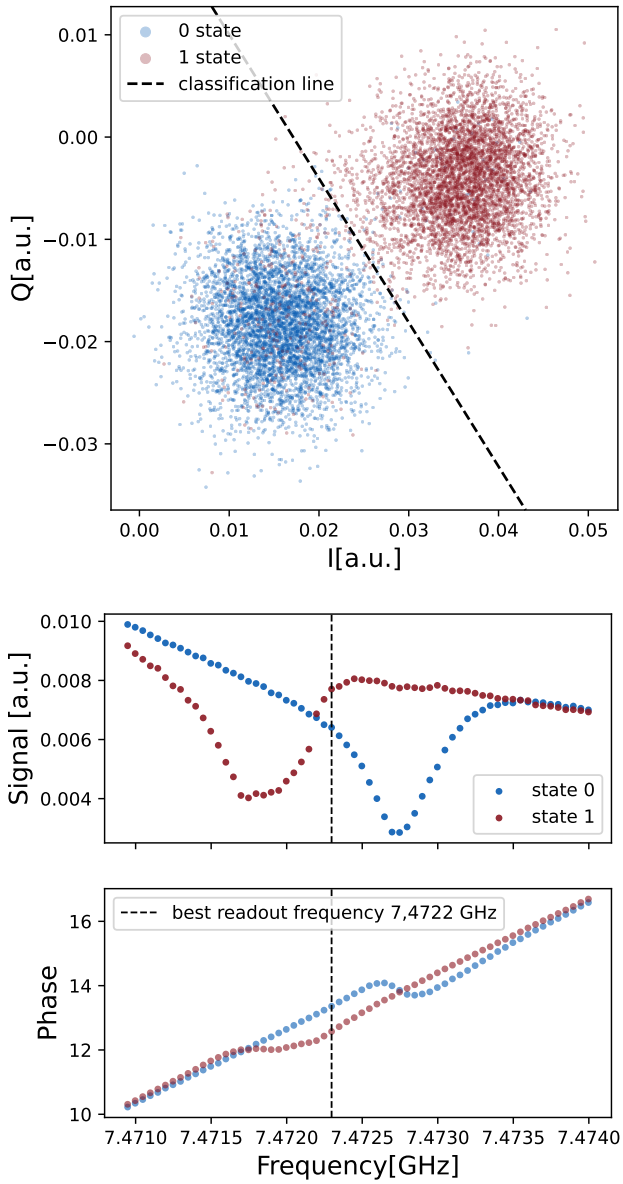


Figure 10.7.: **Top**: Single-shot experiment in which the qubit is prepared first in the ground state (blue dots) and then in the excited state (red dots), and a linear classifier (dashed line) is trained to distinguish between the two. **Middle and bottom**: Measurement of the dispersive shift and determination of the readout frequency that maximizes the separation between ground and excited states in the IQ plane (dashed line).

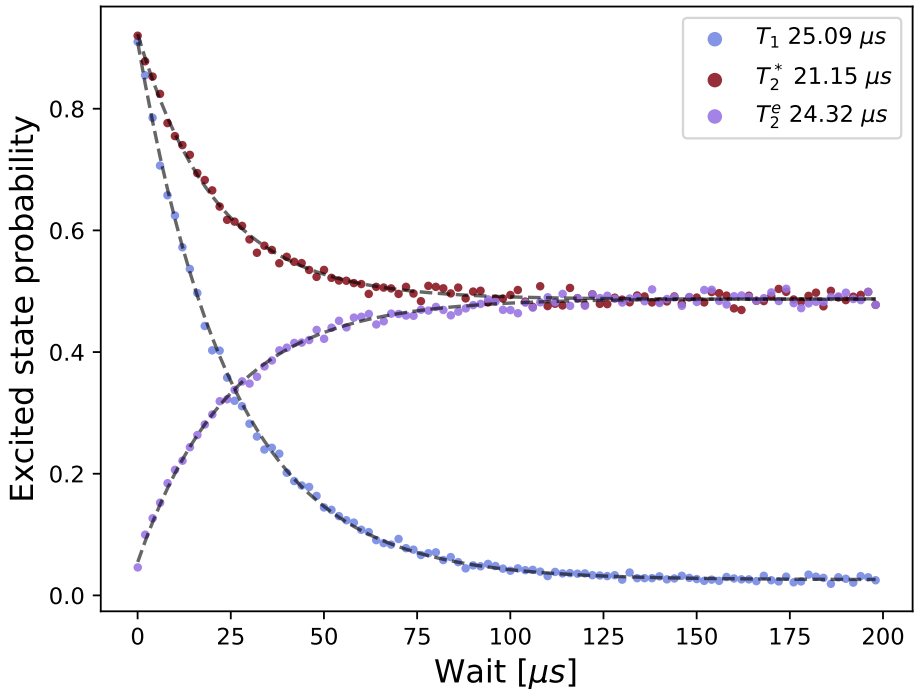


Figure 10.8.: Measurement of the qubit coherence times.

information. It has been shown [53] that the optimal envelope is given by the difference between the average signals corresponding to the two qubit states. Qibocal implements an experiment for extracting this envelope under the name *calibrate state discrimination*.

Readout fidelity can also be enhanced by modifying the shape of the readout pulse, usually rectangular, to reduce the timescale of the resonator field's rise and fall beyond $\frac{1}{\kappa}$. For instance, the initial rise can be accelerated by applying a short but strong pulse [54], while the final decay can be sped up using a fast-emptying sequence inspired by dynamical decoupling techniques [55, 56].

10.3.7. Coherence times

Due to its interaction with the environment, a qubit initially prepared in the excited state $|1\rangle$ undergoes relaxation to the ground state $|0\rangle$, as explained in Sect. 5.1. To characterize this relaxation process, the qubit is first initialized in $|1\rangle$ via a calibrated π-pulse. Subsequently, the population of the excited state is measured after variable waiting times $\Delta\tau$. The temporal evolution of the excited-state population is expected

to follow an exponential decay, which can be modeled as

$$p_e(t) = A + B\, e^{-t/T_1}, \tag{10.9}$$

fitting this formula we can extract T_1.

The acquisition protocol for measuring the coherence time T_2 is analogous to that of the Ramsey experiment, with the primary distinction that in this case the drive pulse is applied on resonance. This protocol assumes that any detuning errors in the drive frequency have already been corrected using a preliminary Ramsey experiment. Consequently, the excited-state population is expected to follow a simple exponential decay:

$$p_e(t) = A + B\, e^{-t/T_2}. \tag{10.10}$$

The reason to disctinct between Ramsey and T_2 experiments lies in the optimization of acquisition times: the Ramsey protocol, which involves a detuned drive, allows for short scans to correct for detuning errors, while the direct T_2 measurement requires longer scans to reliably extract the coherence time.

Another figure of merit of the qubit coherence is the echo coherence time T_2^{Echo}, also the experiment to measure it is similar to the standard Ramsey experiment, with the key difference that a π pulse is applied midway between the two $\pi/2$ pulses.

By including this additional pulse, one can mitigate qubit dephasing [5], since any phase accumulated during the first half of the sequence is effectively reversed during the second half. Consequently, it is generally observed that $T_2^{\text{Echo}} \geq T_2$.

The excited-state population is fitted with a damped exponential of the form:

$$p_e(t) = A + B\, e^{-t/T_2^{\text{Echo}}}. \tag{10.11}$$

The results of this experiments are summarized in Fig. 10.8.

10.3.8. Benchmarking

Within the `Qibocal` library, advanced benchmarking techniques are provided to rigorously evaluate the quality of a calibration experiment. Incorporating benchmarking directly into the `Qibocal` workflow is essential, as it delivers the relevant figures of merit that the calibration routines are designed to optimize.

The simplest experiment to assess the quality of single-qubit gates is the *AllXY* protocol. In this experiment, pairs of single-qubit rotations are applied in sequence, ideally producing a staircase pattern in which the resulting states are restricted to $|0\rangle$, $|1\rangle$, or their superpositions. In practice, different types of imperfections cause systematic deviations from this ideal pattern, leading to characteristic error syndromes [51].

`Qibocal` also includes tomography routines at various levels. While tomographic methods are useful well beyond the context of calibration,

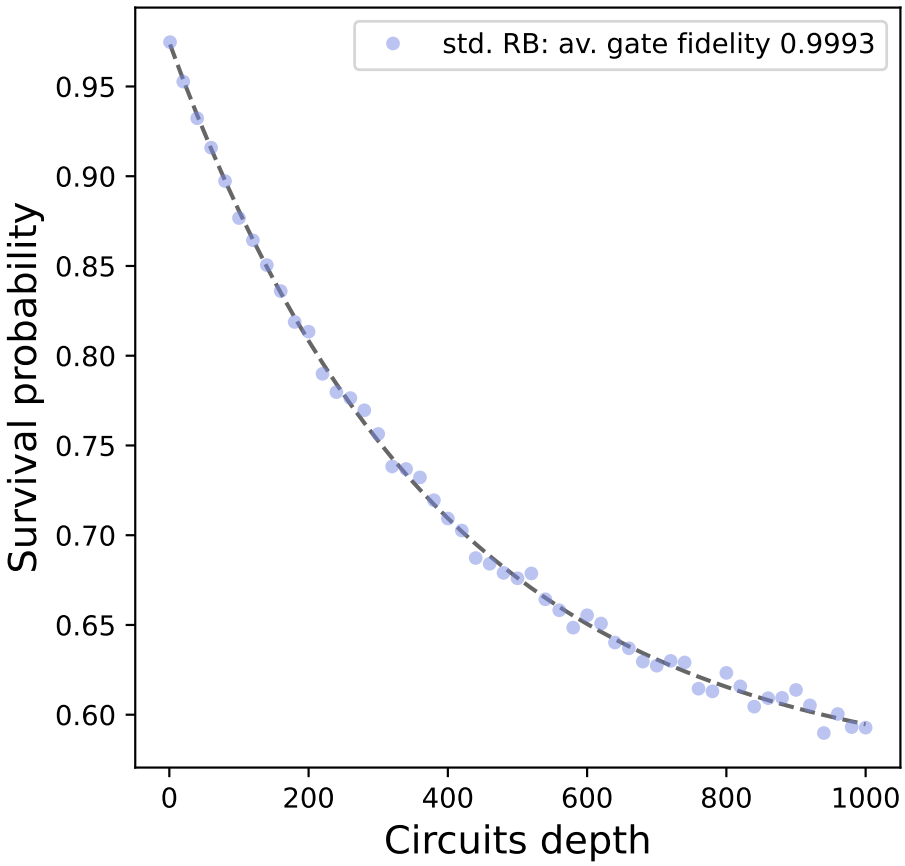


Figure 10.9.: Randomized benchmarks for single qubit gates.

they are particularly valuable here for validating calibrated parameters and extracting detailed information about the resulting quantum states and processes.

The core idea of tomography experiments [57] is to reconstruct the density matrix of the qubit system. For a single qubit, this is achieved by measuring the expectation values of the Pauli operators X, Y, and Z,

$$\rho = \frac{1}{2} + \frac{tr(X\rho)X + tr(Y\rho)Y + tr(Z\rho)Z}{2}, \tag{10.12}$$

therefore three observables are necessary to characterize the density matrix.

A more scalable approach to assess calibration quality is provided by *randomized benchmarking* (RB) [58]. Unlike tomography, which attempts to reconstruct detailed models of each gate, RB evaluates the performance of an entire set of quantum logic gates and summarizes it with $\mathcal{O}(1)$ metrics. These benchmarks report the *average* performance of the gates over many possible input states and contexts, thus offering the end user a practical intuition, though few formal guarantees, about how well the gates will perform within arbitrary circuits.

Randomized benchmarking is often applied to one- or two-qubit gates, but scalable variants exist that can assess the overall performance of an entire processor. While RB provides significantly less detailed and predictive information than tomographic protocols, it is widely used due to its scalability and efficiency.

`Qibocal` implements both *standard* and *interleaved* randomized benchmarking (Fig. 10.9), for single- and two-qubit gates. Further details on these experiments can be found in App. C.

10.4. Two qubit gates calibration

The last step to achieve a working set of universal quantum gates involves the calibration of a two-qubit gate, which requires to eingeneer the interaction between neighbour qubits in order to achieve the desired gate. The type of interaction depends on the hardware, specifically which elements of the interaction (qubits and couplers) are tunable and usually this makes some choices of two-qubit gates easier to calibrate.

The coupling between the two qubits can be direct or mediated by a coupler like a resonator bus or a tunable one like a SQUID connected to a flux line. In all the cases the direct coupling between the different elements could be capacitive or inductive, so it is clear that from the hardware level there are a lot of possible combinations.

In this section, we will go into the details of how to calibrate a CZ gate with a chip with flux tunable qubits and fixed coupling, but `Qibocal` provides also the calibration of the iSWAP gate also in presence of flux tunable couplers and the calibration of CNOT gate via cross resonance.

10.4.1. Qubit-qubit interaction

The system of two interacting qubits is described by the Hamiltonian

$$H = H_{q1} + H_{q2} + H_{int}, \tag{10.13}$$

where H_{qi} is the Hamiltonian of the ith qubit and H_{int} takes into account the interaction between the two qubits. In the classical Hamiltonian, i.e. before the quantization, the interaction term is

$$H_{int} = C_g V_1 V_2, \tag{10.14}$$

where C_g is the coupling capacitance and V_1 (V_2) is the voltage operator of the corresponding voltage node being connected. In the case of inductive coupling, a mutual inductance shared by two loops is the coupling mechanism, yielding an interaction Hamiltonian that is of the intuitive form

$$H_{int} = M_{12} I_1 I_2, \tag{10.15}$$

where M_{12} denotes the mutual inductance between the loops and I_1 and I_2 are the current operators for the currents through the inductors.

In the first case, after the quantization, the Hamiltonian has the form

$$H = H_{q1} + H_{q2} + \hbar J\, (b_1^\dagger b_2 + b_1 b_2^\dagger), \tag{10.16}$$

where

$$H_{qi} = \hbar \omega_{qi} b_i^\dagger b_i - \frac{E_{C_i}}{2} (b_i^\dagger)^2 b_i^2, \tag{10.17}$$

and the interaction amplitude J takes the form

$$\hbar J = \frac{2 E_{C_1} E_{C_2}}{E_{C_c}} \left(\frac{E_{J_1}}{2E_{C_1}} \times \frac{E_{J_2}}{2E_{C_2}} \right)^{\frac{1}{4}}, \tag{10.18}$$

with E_{J_i} and E_{C_i} the transmon Josephson and charging energies, and $E_{C_c} = e^2/2C_c$ the charging energy of the coupling capacitance labelled C_c.

This Hamiltonian describes the coherent exchange of an excitation between the two qubits. In the two-level approximation, assuming $\omega_{q_1} = \omega_{q_2}$, and moving to a frame rotating at the qubit frequency, we get

$$H'_{int} = \hbar J \left(\sigma_1^+ \sigma_2^- + \sigma_1^- \sigma_2^+ \right). \tag{10.19}$$

Evolution under this Hamiltonian for a time $\pi/(2J)$ leads to an entangling iSWAP gate.

To tune the CZ gate we rely on the non anharmonicity and the non-computational states, specifically we move the qubits to the $|11\rangle \leftrightarrow |02\rangle$ avoided crossing, that happens when the difference between the two qubit frequencies is equal to the anharmonicity of the highest frequency

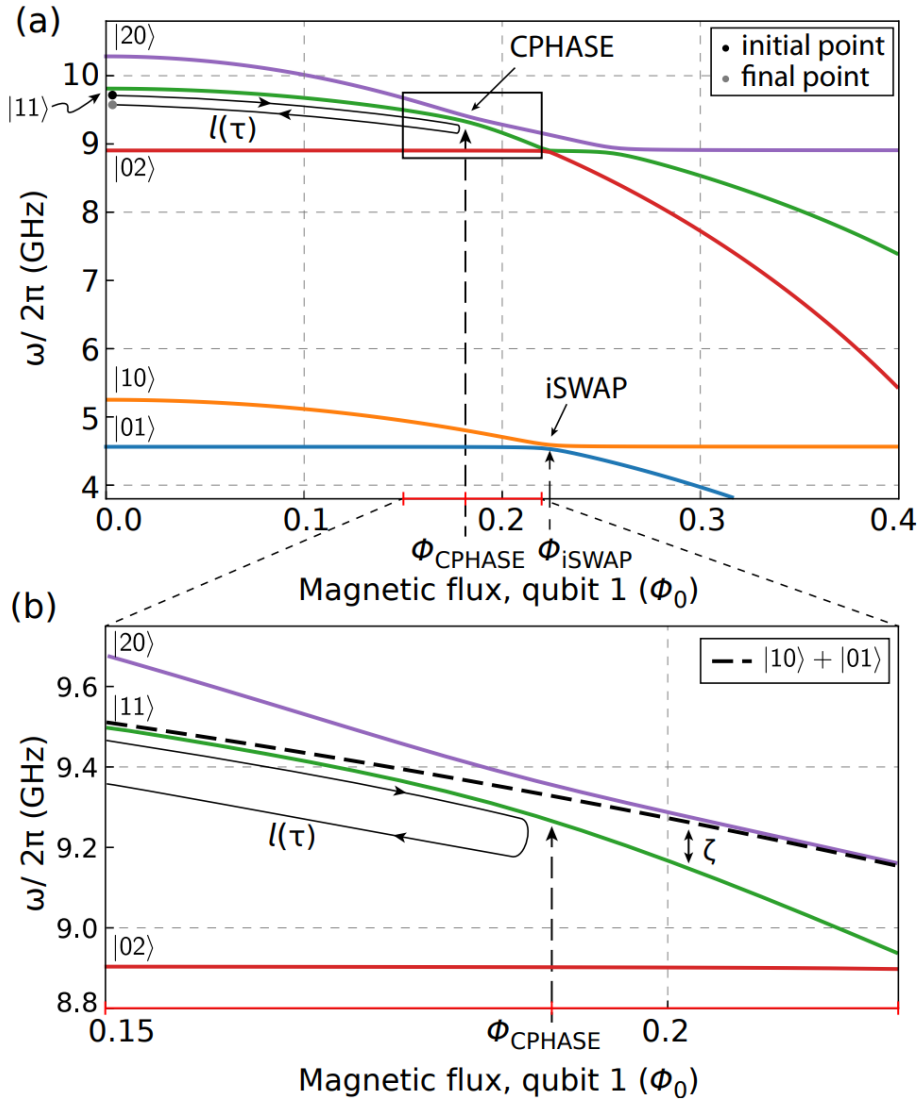


Figure 10.10.: Image taken from [5]. **(a)** Spectrum of two coupled transmon qubits as the local magnetic flux of qubit 1 is varied. The two lower branches, corresponding to $|01\rangle$ and $|10\rangle$, participate in the iSWAP operation. The avoided crossing highlighted by the black rectangle is used to realize the conditional phase (CPHASE) gate at $\Phi = \Phi_{\text{CPHASE}}$. The black line with arrows illustrates a representative trajectory for implementing the CPHASE gate, starting at the black circle and ending at the gray circle. **(b)** Magnified view of the $|20\rangle \leftrightarrow |11\rangle$ avoided crossing marked in panel (a), at $\Phi = \Phi_{\text{CPHASE}}$. The parameter ζ characterizes the energy difference between $|11\rangle$ and $(|01\rangle + |10\rangle)$, while $\ell(\tau)$ represents the trajectory in the (Φ, t)–plane.

one, if we neglect the interaction term. From 10.10, it is clear that we can exploit this interaction thanks to qubit finite negative hanarmonicity, which permits the phase gate to happen before the *i*SWAP.

In the CPHASE interaction point (see Fig. 10.10), the energy E_{11} associated to the state $|11\rangle$ is smaller than the sum of E_{01} and E_{10} by ζ, i.e.,

$$\zeta = E_{11} - E_{01} - E_{10}. \tag{10.20}$$

The adiabatic time evolution of the system corresponds to the unitary operator

$$\hat{CZ}(\phi_{01}, \phi_{10}, \phi_{11}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i\phi_{01}} & 0 & 0 \\ 0 & 0 & e^{i\phi_{10}} & 0 \\ 0 & 0 & 0 & e^{i\phi_{11}} \end{pmatrix}, \tag{10.21}$$

where $\phi_{ab} = \int dt \frac{E_{ab}(t)}{\hbar}$ is the dynamical phase accumulated over the total flux excursion. By applying single qubit gates we can get rid of the extra phases ϕ_{01} and ϕ_{10} and obtain the CZ gate,

$$\hat{CZ}(\phi) = \mathrm{diag}(1,\, 1,\, 1,\, e^{i\phi}) = \hat{R}_Z^1(-\phi_{10})\, \hat{R}_Z^2(-\phi_{01})\, \hat{CZ}(\phi_{01}, \phi_{10}, \phi_{11}), \tag{10.22}$$

with

$$\phi = \phi_{11} - \phi_{01} - \phi_{10} = \int dt\, \zeta(t), \tag{10.23}$$

and where $\hat{R}_Z^i(\theta) = \mathrm{diag}(1,\, e^{i\theta})$ is a single-qubit phase gate acting on qubit i. For $\phi = \pi$ it reduces to the controlled-Z (CZ) gate, for the first time realized in [59].

As pointed out in [5], usually optimal control of adiabatic movement assumes the movement through the avoided crossing but the trajectory in 10.10 moves close to and then back from the avoided crossing. This modification to the adiabatic movement protocol was addressed in [60], specifically in the context of errors for a CPHASE gate implemention. Non-adiabatic errors can be minimal for gate times only slightly longer than $2\pi/g$ using an optimal wave form (based on a Slepian waveform) to parametrize the trajectory.

Another approach is to non-adiabatically bring the system in the 11−02 interaction point and let it evolve for a time t [61]. In this case, the final state is

$$\psi(t) = cos\left(\frac{\zeta t}{2\hbar}\right)|11\rangle + sin\left(\frac{\zeta t}{2\hbar}\right)|02\rangle\,, \tag{10.24}$$

if $t = h/\zeta$ we realize a CZ gate.

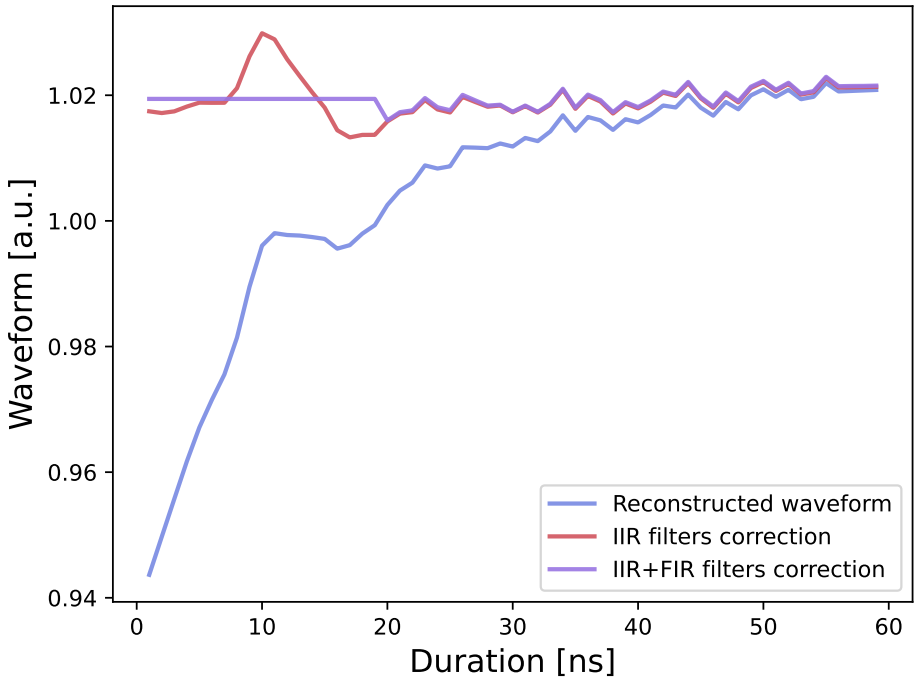


Figure 10.11.: The cryoscope experiment measures and corrects the distortion of flux pulses. The original distorted pulse reaching the qubit (blue) is used to derive IIR (red) and FIR (violet) compensation filters. The final, corrected pulse (violet) closely matches the ideal step function.

10.4.2. Flux pulse distortions

One of the main challenges in implementing two-qubit gates is the presence of distortions in the flux pulses, which originate from the limited bandwidth of the control electronics and transmission lines, and from the bias tees. These distortions not only modify the waveform experienced by the qubit but can also introduce long-timescale effects, where the flux at the qubit at a given time depends on previous flux excursions. A common strategy to mitigate this problem is to pre-distort the pulses according to the measured response of the system.

In this section, we present the procedure for running a Cryoscope experiment [62] using `Qibocal`.

The purpose of the Cryoscope experiment is to reconstruct the actual shape of the flux pulse that reaches the qubit, with the aim of identifying and correcting distortions in the transmitted signal. The method relies on the fact that the transition frequency of a transmon qubit depends on the applied magnetic flux

$$f_Q(\Phi_Q) \approx \frac{1}{h}\left(\sqrt{8E_J E_C\left|\cos\left(\pi\frac{\Phi_Q}{\Phi_0}\right)\right|} - E_C\right), \tag{10.25}$$

where E_C is the charging energy, E_J is the sum of the Josephson energies and Φ_0 is the flux quantum. The routine implementation follows the description given in [62].

The Cryoscope experiment is implemented as a Ramsey-like sequence, in which a flux pulse is inserted between two $\pi/2$ pulses separated by a fixed time interval T. The first $\pi/2$ rotation around the Y axis prepares the qubit in the superposition state $|0\rangle \rightarrow (|0\rangle + |1\rangle/\sqrt{2})$. Subsequently, the flux pulse modifies the phase of the superposition, resulting in the state $(|0\rangle + e^{i\phi_\tau}|1\rangle/\sqrt{2})$, where ϕ_τ denotes the phase accumulated during the flux pulse of duration τ,

$$\frac{\phi_\tau}{2\pi} = \int_0^T \Delta f_Q(\Phi_{Q,\tau}(t))\, dt. \tag{10.26}$$

Finally, the sequence is completed with a $\pi/2$ rotation applied either around the Y axis or around the X axis, allowing the measurement of the corresponding $\langle X\rangle$ or $\langle Y\rangle$ component of the Bloch vector, respectively. From the combined values of $\langle X\rangle$ and $\langle Y\rangle$, one can reconstruct the relative phase ϕ_τ, which in turn provides direct access to the time-dependent frequency shift of the qubit during the flux pulse.

From the measurements of $\phi_{\tau+\Delta\tau}$ and ϕ_τ, the relative frequency shift Δf_R can be computed as

$$\Delta f_R = \frac{\phi_{\tau+\Delta\tau} - \phi_\tau}{2\pi\Delta\tau} = \frac{1}{\Delta\tau}\int_\tau^{\tau+\Delta\tau} \Delta f_Q\big(\Phi_{Q,\tau+\Delta\tau}(t)\big)\, dt + \varepsilon, \tag{10.27}$$

with

$$\varepsilon = \frac{1}{\Delta\tau}\left(\int_{\tau+\Delta\tau}^{T_{\text{sep}}} \Delta f_Q\big(\Phi_{Q,\tau+\Delta\tau}(t)\big)\, dt - \int_\tau^{T_{\text{sep}}} \Delta f_Q\big(\Phi_{Q,\tau}(t)\big)\, dt\right). \tag{10.28}$$

Finally, we can extract an estimate of the effective flux pulse $\Phi_Q(t)$ on the qubit by inverting Eq. 10.25.

The phase contribution arising from turn-off transients is negligible, because of the sharp return to the first-order flux-insensitive sweet spot of the nearly quadratic dependence $\Delta f_Q(\Phi_Q)$. Numerical simulations indicate that the ratio $|\varepsilon|/\Delta f_R$ typically lies in the range of 10^{-2} to 10^{-3} for realistic dynamical distortions introduced by commonly used electronic components. Consequently, this contribution will be neglected.

The objective of the Cryoscope protocol is to determine the appropriate predistortion of the flux pulse required to generate the desired detuning on the qubit. This predistortion is implemented by filtering the waveform produced by the AWG through a combination of finite impulse response

(FIR) and infinite impulse response (IIR) digital filters. Infinite impulse response (IIR) filters are a class of digital filters that make use of both past input samples and feedback from previous output values. In discrete time, the general form of an IIR filter can be expressed through the equation:

$$a_0 y[n] = \sum_{i=0}^{N} b_i x[n-i] - \sum_{i=1}^{M} a_i y[n-i], \tag{10.29}$$

where $x[n]$ is the input signal, $y[n]$ is the output signal, b_i are the feedforward coefficients, a_i are the feedback coefficients, and N and M denote the orders of the feedforward and feedback components, respectively.

This recursive structure enables IIR filters to approximate systems with exponential or resonant dynamics, making them particularly effective for correcting physical distortions such as exponential overshoot or undershoot. To determine the first-order IIR filter, the step response is fitted with the formula

$$s(t) = g(1 + Ae^{-\frac{t}{\tau_{IIR}}})u(t), \tag{10.30}$$

where τ_{IIR} is the time constant, A is the amplitude, g is a gain correction factor that can be ignored, and $u(t)$ is a rectangular pulse. From these coefficients, all the filter taps can be extrapolated.

After determining the feedback and feedforward coefficients of the IIR filter, it is necessary to implement a finite impulse response (FIR) filter in order to compensate for residual distortions on shorter timescales (typically less than 30 ns), which are not fully captured by the IIR correction. An FIR filter generates its output as a weighted sum of finite number of past input samples. Its general form is given by

$$y[n] = \sum_{i=0}^{N} b_i x[n-i], \tag{10.31}$$

where b_i are the filter coefficients and N is the order of the FIR filter.

The coefficients $\{b_i\}$ of the FIR filter are determined by minimizing the average relative deviation between the filtered output obtained by applying the FIR filter to the signal already corrected by the IIR filter and the corresponding ideal step response.

The cryoscope experiment can compensate for short time scales at the order of the qubit coherence time, but the pulse distortions can happen at longer time scales reducing the gates repeatibility.

In `Qibocal` we provide the protocol described in [63], correcting the flux pulse up to tens of microseconds, see Fig. 10.12. In this experiment after fluxing the qubit away from its sweetspot, in order to be more sensitive to flux changes, we wait a variable time τ before driving the qubit. Through a spectroscopy on the drive line we are able to measure the qubit frequency. Due to the bias-tee, we expect to observe an exponential rise to the correct frequency value. In the protocol, after

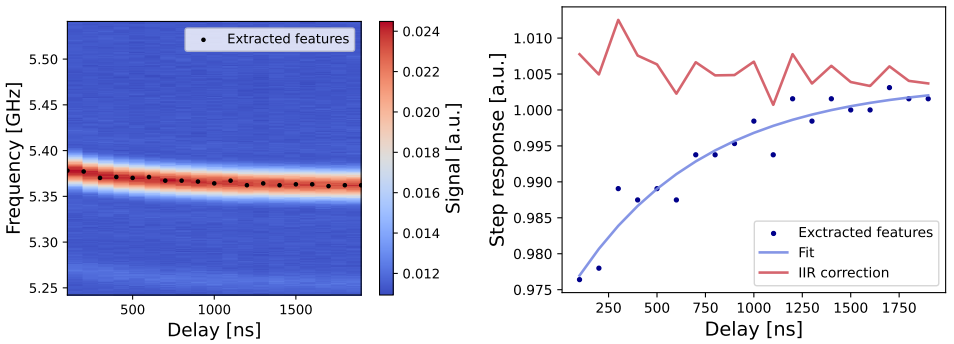


Figure 10.12.: This measurement characterizes flux distortions over tens of microseconds and demonstrates their correction. The pulse shape is first reconstructed via a two-tone experiment (left). Then, a series of IIR filters are applied to compensate for the distortion, producing the corrected pulse (right, red line).

performing an exponential fit, we compute the corresponding IIR filter to correct it.

10.4.3. CZ calibration

At the beginnign of the two qubits calibration with non adiabatic pulses, the proper flux pulse amplitude and duration should be find, in order to bring the qubit pair into the interaction point and let them evolve for the proper amount of time. This pulse parameters are usually tuned with the Chevron experiment, both for CZ and *i*SWAP gates. At this level of the calibration we are using rectangular pulses.

For the iSWAP gate, the calibration sequence consists of a π pulse followed by a flux pulse with variable amplitude and duration, applied to the qubit with the higher frequency. The initial π pulse excites the qubit to the state $|1\rangle$, after which the flux pulse detunes its frequency close to resonance with the second qubit. The iSWAP interaction is realized by exploiting the avoided crossing between the states $|10\rangle$ and $|01\rangle$.

The resulting population dynamics are expected to follow the oscillatory behavior

$$p_e(t,\Delta) = \frac{\Delta^2}{\Delta^2+4g^2} + \frac{4g^2}{\Delta^2+4g^2}\cos^2\left(\frac{\sqrt{\Delta^2+4g^2}}{2}t\right), \tag{10.32}$$

where $\Delta = \omega_1 - \omega_2$ is the detuning between the two qubits and g is their coupling constant.

When this expression is measured as a function of both the pulse duration t and the detuning Δ, the resulting excitation probability map

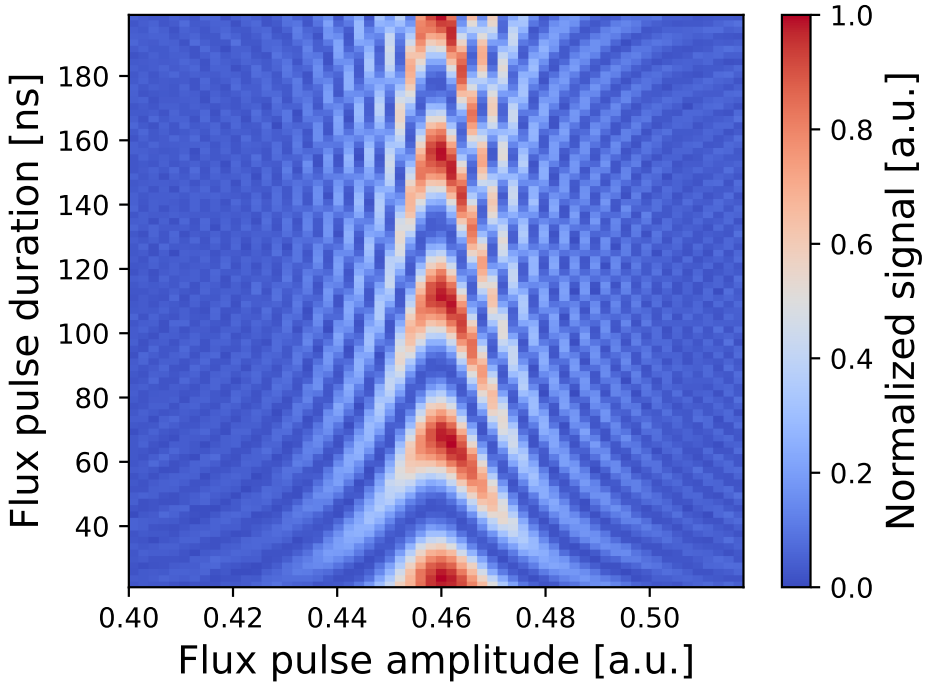


Figure 10.13.: We characterize the flux pulse's duration and amplitude to implement a CZ interaction between qubits. The chevron pattern emerges by scanning these parameters, revealing the precise conditions needed for the CZ gate.

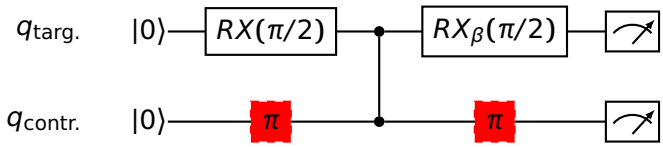


Figure 10.14.: Circuit representation of the conditional oscillation experiment.

exhibits a characteristic "Chevron" pattern. Closer to resonance the frequency of the finges decreases and the amplitude increases, reflecting the hybridization of the states $|10\rangle$ and $|01\rangle$. By analyzing the position of the central fringe and the oscillation frequency at resonance, it is possible to extract both the optimal flux bias and the effective coupling strength g, which are required to calibrate the iSWAP gate.

The calibration of the CZ gate employs a similar sequence, with the addition of a π pulse applied to the lower-frequency qubit so that both qubits are initially prepared in the state $|11\rangle$. In this case, the gate operation relies on the avoided crossing between the states $|11\rangle$ and $|20\rangle$, thereby enabling the conditional phase accumulation required for the CZ gate.

To conclude the calibration, we have to correct for the extra phases that we have accumulated during the flux pulses, as explained in 10.4.1. The extra phases evaluation is done through the conditional oscillation experiments where we execute two circuits. In the first one, the control qubit remains in the ground state while the target qubit, prepared in a superposition state by a $\pi/2$ pulse, undergoes the CZ gate and a subsequent $\pi/2$ pulse with variable phase β before measurement, i.e., a rotation on the Z axis of angle β before the gate execution, as depicted in Fig. 10.14. The probability of the target qubit to be in the $|1\rangle$ state is

$$p(\beta) = \frac{1}{2} + \frac{1}{2} cos(\beta + \phi_{10}). \tag{10.33}$$

The second one follows an identical sequence for the target but initializes the control qubit in its excited state, the measured probability of the target qubit is

$$p(\beta') = \frac{1}{2} + \frac{1}{2} cos(\beta' + \phi_{11} - \phi_{01}) \tag{10.34}$$

From this, we can evaluate the extra phase ϕ_{10} accumulated by the CZ gate and from the difference between the phases of the two probabilities we can extrapolate the angle of the phase gate ϕ, as defined in eq. 10.23.

Furthermore, the reduction in oscillation amplitude, quantified as the missing fraction m, provides a rapid leakage estimate via $\widetilde{L_1} = m/2$,

where m is the difference between the probabilities of the control qubits in the two cases as explained in [64].

In Fig. 10.16 we show the standard and interleaved randomized benchmarking obtained by calibrating a CZ gate with an adiabatic flat-top gaussian pulse after the calibration procedure described in this section. The performances of the CZ degrades quickly, probably because of remaining long-pulse distortions.

10.4.4. SNZ calibration

The implementation of the *CPHASE* gate with an unipolar flux pulse suffers from two main issues: the first-order sensitivity to flux noise due to the detuning of the qubit from the sweet spot, and distortions in the flux lines, especially in the long-timescale regime.

A possible solution is the Net-Zero (NZ) [65] pulse (Fig. 10.18), which uses a pair of opposing, short unipolar flux pulses that sum to zero. By eliminating the DC component of the pulse, the new *CZ* gate is resilient to long-timescale distortions and their resulting history-dependent errors, enhancing the gate's repeatability. This design cancels out leakage errors because the derivative of the flux arc, $d\Phi/dt$, is equal in magnitude and opposite in sign at the positive and negative halves of the NZ pulse.

Building on this, the Sudden Net-Zero (SNZ) [64] pulse is a simplified variant. As the name suggests, this pulse also has a zero average, but in contrast to the NZ pulse, it uses square pulses of amplitude A instead of adiabatic ones, with an holding interval of duration t_ϕ in the middle (10.18). This simple choice conserves all the advantages of the NZ pulse while making the pulse easier to calibrate, as it requires only two parameters for its definition. The SNZ also offers the possibility to operate at the speed limit of the coupling g. Furthermore, the landscape of the conditional phase and leakage has a regular structure, see Fig. 10.17. The leakage landscape exhibits a clear structure featuring a vertical leakage valley at $A = 1$ in the simulation, while looking at the phase landscape, It is clear that the region with the π phase is the one connecting the two areas. with the highest leakage. From the intersection of these two trajectories, it is easy to find the best parameters. Looking at Fig. 10.17, it is clear that the idling period is not needed in the idealized scenario with infinite time resolution and no decoherence, but it is important to include t_ϕ in experiment, since any flux-pulse distortion remaining from the first half pulse during the second (e.g., due to finite pulse rise time) will break the symmetry between the two rectangular pulses.

The first step for the calibration of the SNZ pulse is evaluating the resonance amplitude A and the speed limit ($t_{lim} = \pi/g$) for the $|11\rangle - |02\rangle$ interaction, fundamentally constrained by the coupling strength. These parameters are extracted from the characteristic chevron pattern of a unipolar square flux pulse; the symmetry axis of this pattern indicates the resonance amplitude, while the oscillation period along this axis reveals t_{lim}, as explained in 10.4.3.

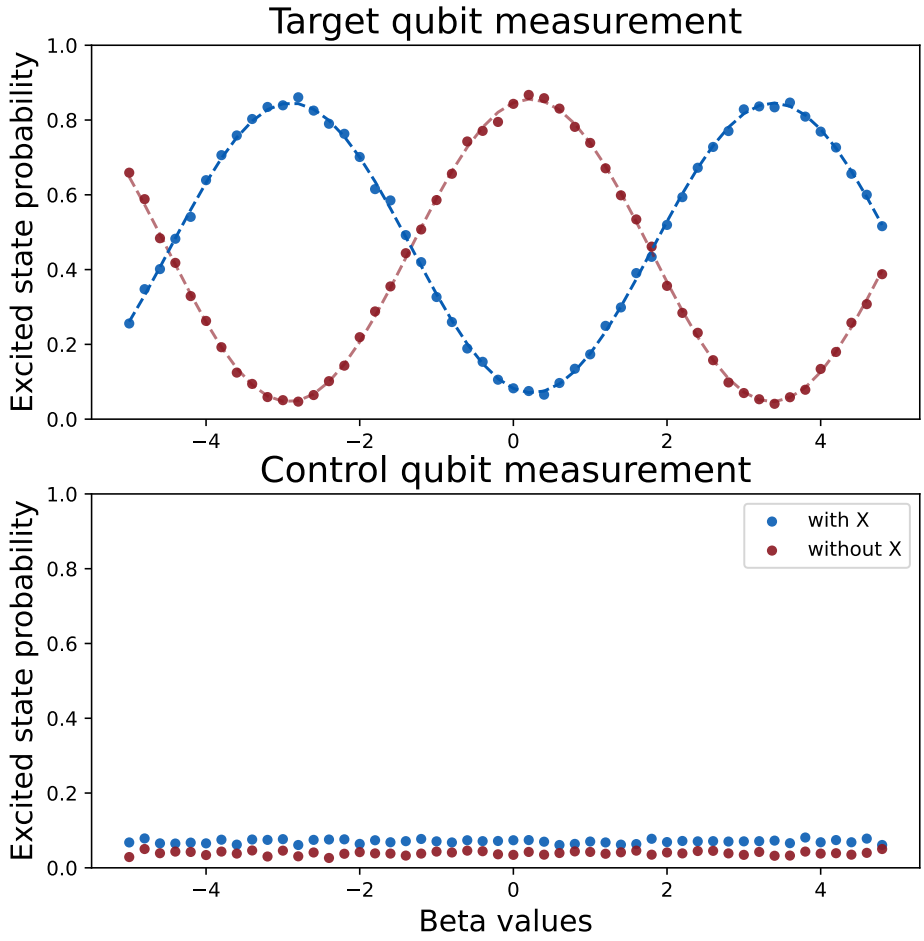


Figure 10.15.: Results of conditional phase experiment with SNZ pulse. In this case the angle of the phase gate is 3.11 with a leakage of 1.5%.

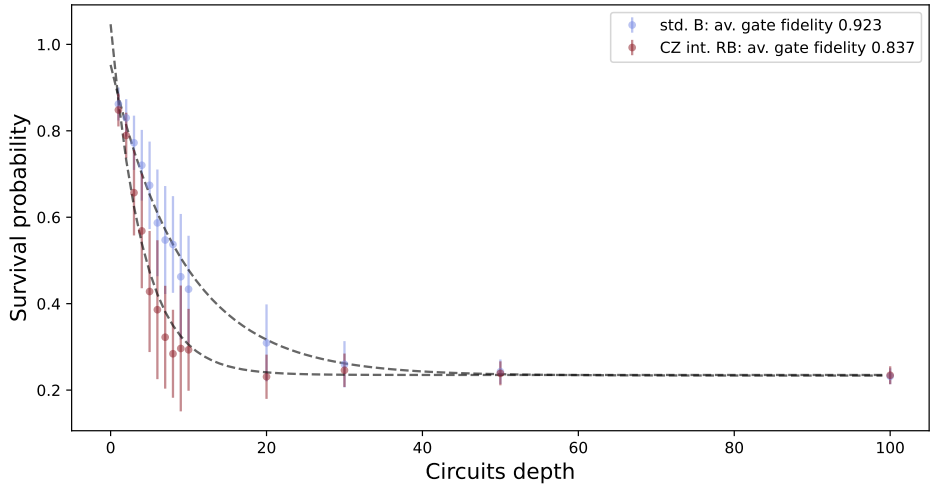


Figure 10.16.: Standard and interleaved two-qubits RB with adiabatic pulse.

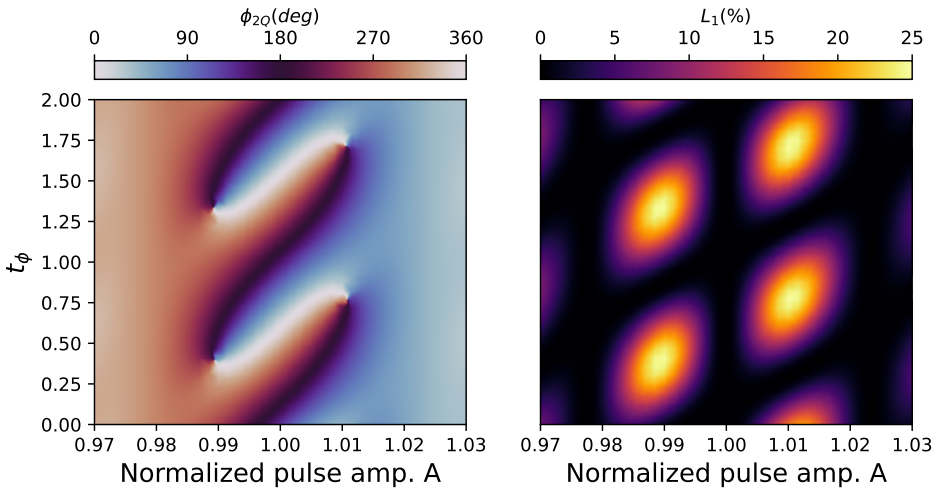


Figure 10.17.: Simulated leakage and gate angle landscape of the SNZ pulse. The data are taken from [64].

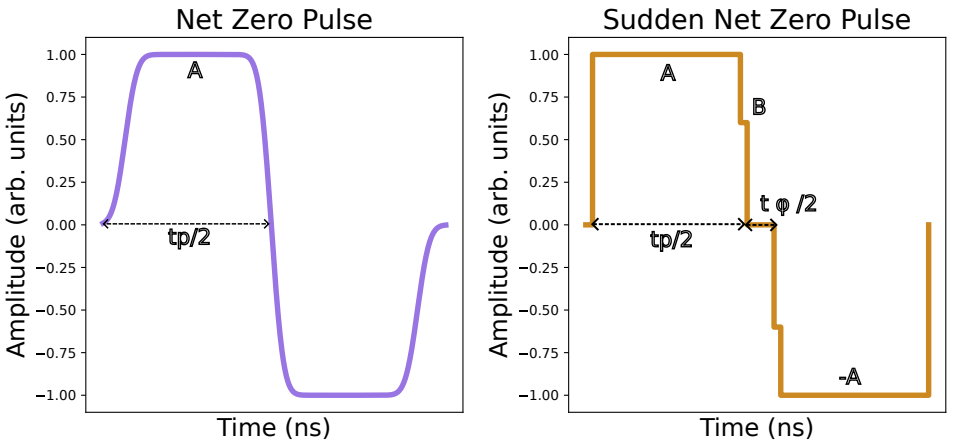


Figure 10.18.: The NZ (left) and SNZ (right) pulse shapes.

Subsequently, the total duration of the SNZ pulse (t_p) is set as close as possible to, but not shorter than, t_{lim}, accounting for the AWG's discrete sampling period t_s. An intentional idling period t_ϕ is then introduced between the two half pulses of the SNZ gate to permit the precise accumulation of a relative phase ϕ between the $|02\rangle$ and $|11\rangle$ states. To mitigate the negative effects of choosing only discretized values of t_ϕ, an additional step of amplitude B is added at the end of the first unipolar pulse and at the beginning of the second one, as shown in Fig. 10.18. The core of the calibration involves the measurement of the conditional phase ϕ_{2Q} and an estimated leakage L_1 as functions of the main pulse amplitude A and a fine-tuning amplitude B using conditional-oscillation experiments, as in Fig. 10.19

By superimposing the $\phi_{2Q} = 180°$ contour coming from the phase landscape onto the leakage one, optimal control parameters (A, B) can be pinpointed at the intersections where the phase condition is met within a low leakage area. This regular and predictable structure significantly simplifies the search for high-fidelity, low-leakage gate parameters.

Finally, after optimizing the SNZ pulse parameters, we have to apply the Z-axis rotations to null any accumulated single-qubit phases, thereby completing the CZ gate. Depending on the specific qubit pair and processor architecture, additional steps such as applying parking flux pulses to spectator qubits or selecting an alternative interaction pathway may be necessary to mitigate unwanted couplings or transient phenomena. The quality of the CZ calibration is assesed through standard and interleaved randomized benchmarking (Fig. 10.20).

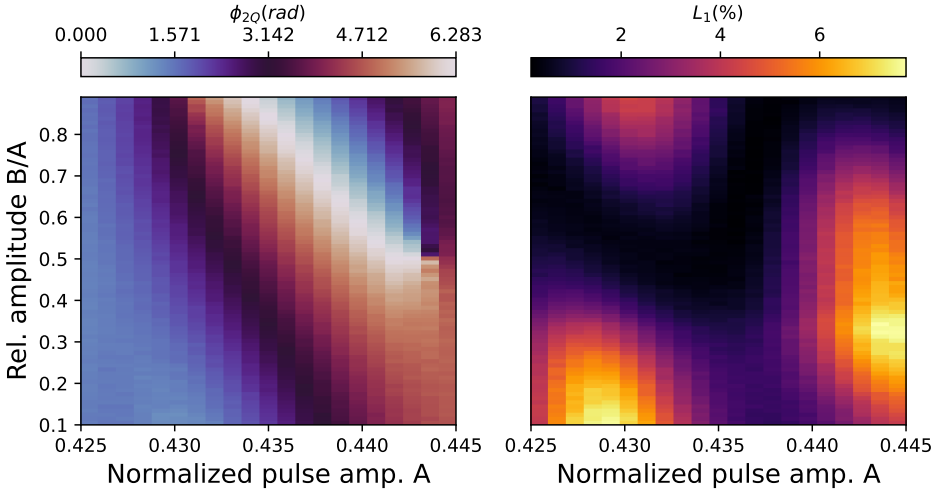


Figure 10.19.: Two-dimensional sweep of the SNZ pulse parameters A and B, characterizing the gate angle (left) and leakage (right). The optimal operating point is selected where the gate angle approaches π with negligible leakage.

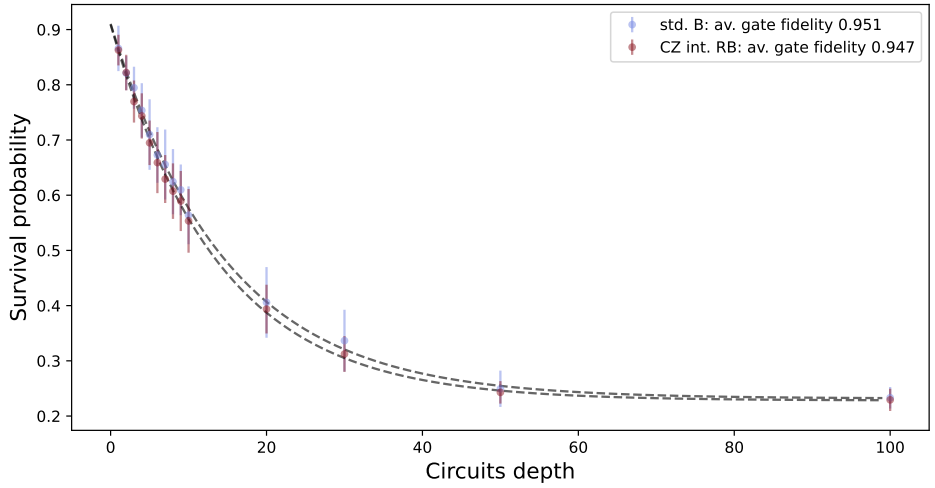


Figure 10.20.: Standard and interleaved randomized benchmarks for two qubit gates with SNZ pulses.

11. Double bracket quantum algorithm

11.1. Introduction

Estimating the ground state energy of a quantum Hamiltonian remains a significant challenge. While quantum phase estimation (QPE) [66, 67, 68, 69, 70] is anticipated to be the standard method in the fault-tolerant era, its demanding circuit depth makes it infeasible for near-term quantum hardware. Consequently, the variational quantum eigensolver (VQE) [71] has become a leading near-term alternative [72, 73, 74]. Nevertheless, as a heuristic algorithm that navigates a non-convex landscape, VQE lacks convergence guarantees and is prone to serious optimization difficulties [75, 76, 77, 78, 79, 80].

Double-bracket quantum algorithms (DBQAs) have recently emerged as a recursive method for constructing Hamiltonian-diagonalizing circuits [81]. This chapter explores their potential for ground state preparation. A key advantage of DBQAs is their suitability for near-term hardware, as they require no auxiliary qubits, unlike quantum phase estimation. Furthermore, and in sharp contrast to VQE, they feature guaranteed analytical convergence [82, 83, 84], using classical optimization solely to enhance performance rather than being essential for the algorithm's operation.

As Fig. 11.1 illustrates, we propose a two-stage protocol for ground state preparation: an initial stage using a short-depth circuit to generate a rough approximation, followed by a DBQA stage to refine it. We use VQE for initialization to demonstrate that DBQA can improve even a low-quality starting state, as might be produced on near-term hardware. We emphasize that our protocol is agnostic to the specific initialization method; other suitable candidates include Hartree-Fock circuits [85], quantum imaginary-time evolution [86, 87, 88, 89], and qubitization techniques using auxiliary qubits [66, 67, 68].

We demonstrate our method's efficacy through numerical simulations of DBQA applied to a lattice Heisenberg model. Our analysis focuses on the relationship between the initial state's quality and the DBQA circuit depth required for convergence. For all tested VQE ansatze and target Hamiltonians, we find that a single DBQA step, with a circuit depth of 50–100 CZ gates per qubit, can improve the energy estimate by an order of magnitude, provided the initial state's energy is closer to the ground state than to the first excited state.

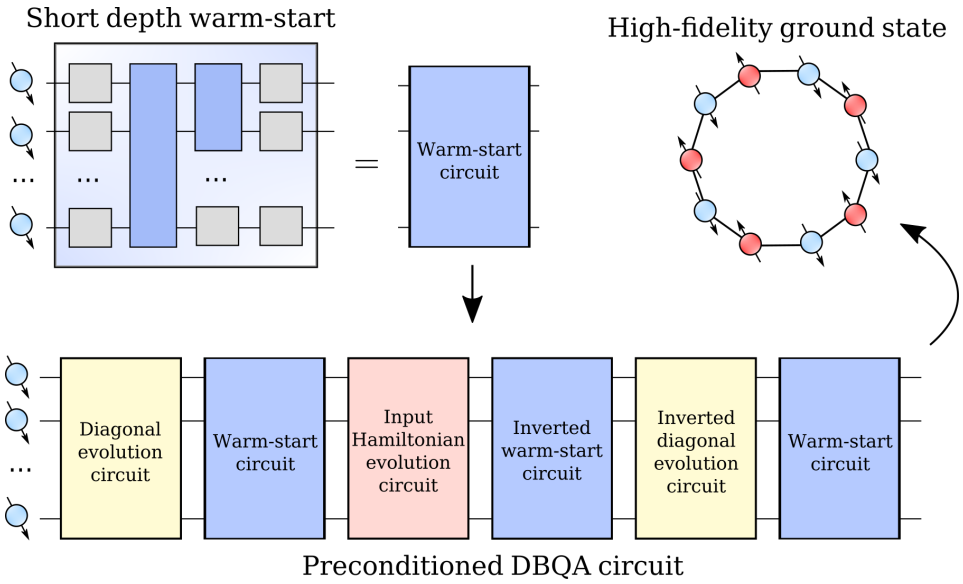


Figure 11.1.: We propose a two-stage ground state preparation protocol: first, apply a relatively short-depth warm-start circuit; second, apply a DBQA circuit to further the ground state preparation fidelity.

11.2. Warm-starting DBQA

DBQAs are quantum protocols inspired by flow equations originally used to study quantum many-body systems [90, 91, 92]. As defined in Ref. [81], DBQA circuits are obtained recursively from double-bracket iterations (DBIs),

$$\hat{A}_{k+1} = e^{s_k \hat{W}_k} \hat{A}_k e^{-s_k \hat{W}_k} \tag{11.1}$$

where $\hat{A}_0$ is the DBI input and the DBI generators are commutators $\hat{W}_k := [\hat{D}_k, \hat{A}_k]$ of $\hat{A}_k$ with operators $\hat{D}_k$ which should be taken diagonal to target diagonalization [82, 81].

To understand the DBI circuit ansatz (detailed in App. D.1), three key points are essential. First, the operators $\hat{R}_k = e^{-s_k \hat{W}_k}$ are unitary provided $\hat{D}_k$ and $\hat{A}_k$ are Hermitian and $s_k \in \mathbb{R}$. Second, the algorithm can be initialized with $\hat{A}_0$ as an input Hamiltonian $\hat{H}_0$, a strategy explored in Ref. [81]. Third, a primary application of DBIs is Hamiltonian diagonalization.

Under specific conditions, when all $\hat{D}_k$ are equal and non-degenerate and the s_k are sufficiently small, $\hat{A}_k$ converges exponentially to a diagonal fixed-point $\hat{A}^{(\infty)}$ [82, 83, 84, 93]. Consequently, the state $|\psi_\infty\rangle := \lim_{k\to\infty} \hat{R}_0 \hat{R}_1 \ldots \hat{R}_k |0\rangle$ becomes an exact eigenstate of $\hat{A}_0$. In the general case, convergence may require circuit depths exponential in k [81] and can yield any eigenstate. Nevertheless, we demonstrate that

even a few iterations can yield substantial improvements.

The performance of DBIs is governed by the step durations s_k and the choice of diagonal operators $\hat{D}_k$. These parameters can be optimized to enhance the diagonalization effect in each step, as measured by a specific cost function. For instance, Ref. [81] selects parameters to minimize the magnitude of the off-diagonal elements in $\hat{A}_{k+1}$ after applying the update in Eq. (11.1)

In this chapter, we address the problem of preparing the ground state of an input Hamiltonian $\hat{H}_0$. To do this, we consider a warm-start unitary $\hat{Q}$ by setting

$$\hat{A}_0 = \hat{Q}^\dagger \hat{H}_0 \hat{Q}. \tag{11.2}$$

This mechanism bridges the initial state approximation (which provides $\hat{Q}$) and our DBQA protocol. We then define our cost function as the average energy $\langle\psi_k|\hat{H}|\psi_k\rangle$ of the states $|\psi_k\rangle := \hat{Q}\hat{R}_0 \ldots \hat{R}_k|0\rangle$ at each iteration, where $\hat{H}$ is the input Hamiltonian,

$$E^{(k)} := \langle\psi_k|\,\hat{H}_0\,|\psi_k\rangle = \langle 0|\hat{A}_{k+1}|0\rangle\,. \tag{11.3}$$

Thus this warm-start mechanism allows us to interface VQE and DBQA by defining a common cost function.

VQE [71] is a variational algorithm that minimizes the energy expectation value $\langle 0|\hat{U}_\theta^\dagger \hat{H}_0 \hat{U}_\theta|0\rangle$ by optimizing parameters θ of a quantum circuit $\hat{U}_\theta$. However, VQE is limited by trainability and expressibility issues [76, 79, 94, 73, 75, 80]. Since DBQA only requires a rough ground state approximation, VQE can still serve effectively as a warm start. We therefore set $\hat{Q} = \hat{U}_{\theta^*}$ in Eq. (11.2) and refer to this combined approach as VQE*x*DBQA.

While Ref. [81] established DBQAs as effective for eigenstate preparation via their proven convergence [82, 84], it did not address how to specifically target the ground state. As our results show, the VQE*x*DBQA protocol consistently lowers the energy. We attribute this to two factors: the Hamiltonian's spectral gap and DBQA's exponential convergence. In its later stages, DBQA turns $\hat{A}_k$ into a mostly diagonal operator, making the commutator $[\hat{D}_k, \hat{A}_k]$ small. Consequently, the unitaries $\hat{R}_k$ cannot induce energy changes larger than the spectral gap. The state must therefore rapidly converge to the nearest eigenstate. When initialized by a method like VQE that is closer to the ground state than to any excited state, $|\psi_\infty\rangle$ becomes the ground state.

11.3. Compiling DBQAs

To execute DBQA circuits, an explicit strategy for compiling the DBI unitaries $\hat{R}_k$ as quantum circuits is necessary. This can be achieved

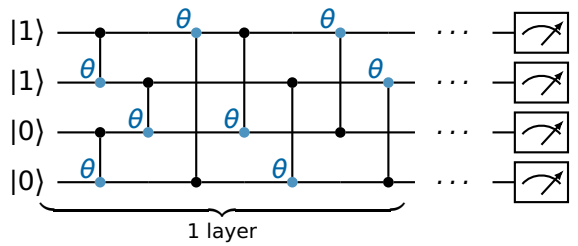


Figure 11.2.: Hamming-weight-preserving architecture for $L = 4$ qubits and $S = 2$. A single circuit layer consists of Reconfigurable Beam Splitter (RBS) gates (see Eq. (11.6)) connecting nearest- and next-nearest neighbors. We account for periodic boundary conditions by adding RBS gates connecting the last and first qubits. After a chosen number of layers, we add measurements in the computational basis.

by *group commutator iterations* (GCIs), where we replace $\hat{R}_k$ by an approximation derived in Ref. [95],

$$\hat{V}_k = e^{ir_k\phi\hat{D}_k} e^{ir_k\hat{A}_k} e^{-ir_k(\phi+1)\hat{D}_k} e^{-ir_k(1-\phi)\hat{A}_k} e^{ir_k\hat{D}_k}, \tag{11.4}$$

with $\phi = \frac{1}{2}(\sqrt{5}-1)$. We then define the unitary of k GCI steps as

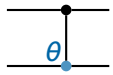
$$\hat{U}_k = \hat{V}_0\hat{V}_1 \dots \hat{V}_{k-1}, \tag{11.5}$$

such that now $\hat{A}_k = \hat{U}_k^\dagger \hat{A}_0 \hat{U}_k$. Following Ref. [95, Eq. (8)], for $r_k = \sqrt{s_k}$ this gives $\hat{V}_k^\dagger \hat{A}_k \hat{V}_k \approx \hat{R}_k^\dagger \hat{A}_k \hat{R}_k$, with an error bounded as $O(s_k^2)$. See App. D.2 for an overview of other group commutator approximations and scalings [96, 81, 95]. The GCI unitary can be recursively compiled using the identity $e^{ir_k\hat{A}_k} = \hat{U}_k^\dagger \hat{U}_{\theta^*}^\dagger e^{ir_k\hat{H}_0} \hat{U}_{\theta^*} \hat{U}_k$ and $\hat{U}_{k+1} = \hat{U}_k\hat{V}_k$ and by calling Hamiltonian simulation [97, 98] for each appearance of an evolution operator in $\hat{V}_k$, see App. D.2 for more details. This recursive unfolding for each step k leads to a circuit depth that grows exponentially with k so we use DBQA for up to three steps and after a warm-start.

11.4. Tailoring VQE

We present quantitative results for ground state preparation of the XXZ model, which exhibits total-spin conservation [99]. For an even number of qubits L, the ground state lies in the half-filling subspace (total spin $S = L/2$). To respect this symmetry, we employ a Hamming-weight-preserving VQE ansatz [100, 101, 102, 103, 104, 105, 106, 107, 108, 79, 80, 109, 110]. This confines the search to a Hilbert space of size

$\binom{L}{L/2}$), yielding polynomial compression. Although not exponential, this restriction to the relevant subspace accelerates training [107, 111, 112, 113, 114]. One method to construct such an ansatz uses a network of Reconfigurable Beam Splitter (RBS) gates.



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (11.6)$$

The Reconfigurable Beam Splitter (RBS) gate is a standard component for building Hamming-weight-preserving circuits. Figure 11.2 shows the specific architecture used in our simulations. For comparison, we also present results in App. D.4 using a hardware-efficient ansatz [115], which searches the full 2^L-dimensional Hilbert space. While this greater expressibility makes training more difficult, interfacing with DBQA again proves highly beneficial.

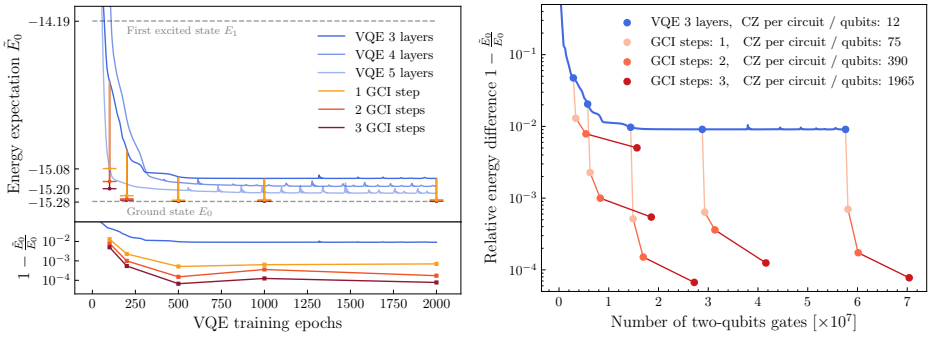


Figure 11.3.: Visualization of the impact of VQExDBQA on cost function for a single VQE random seed, see Tab. 11.1 for statistical analysis. *(left)* Training of VQE (blue lines) for 3, 4, and 5 layers of Hamming-weight preserving ansatz (hues of blue) achieves within 500 training epochs ground state energy residue $\Delta E \approx 1\%$. For more epochs, the initially rapid decrease in the cost function saturates and shows marginal improvement afterward. We initialize DBQA with VQE for selected epochs $\in \{100, 200, 500, 1000, 2000\}$ and optimize DBQA parameters with CMA-ES [116]. In the bottom panel we show the relative difference value between the achieved energy $\tilde{E}_0$ and the true ground state energy E_0. *(right)* Token cost estimates of VQExDBQA by counting the total number of CZ gates required for the complete protocol: training the VQE until a target epoch and the optimization of DBQA.

Layers	**Warm-start**		**1 GCI step**		**2 GCI steps**		**1 DBI step**	**Long VQE training**	
	$1-\tilde{E}_0/E_0$		$1-\tilde{E}_0/E_0$		$1-\tilde{E}_0/E_0$		$1-\tilde{E}_0/E_0$	$1-\tilde{E}_0/E_0$	
3	0.012 ± 0.004		0.0011 ± 0.0007		0.0005 ± 0.0004		0.0009 ± 0.0006	0.010 ± 0.004	
4	0.008 ± 0.004		0.0006 ± 0.0005		0.0002 ± 0.0002		0.0005 ± 0.0004	0.004 ± 0.002	
5	0.005 ± 0.003		0.0003 ± 0.0002		0.0001 ± 0.0001		0.0002 ± 0.0002	0.003 ± 0.002	
	Depth	Cumulative cost	Depth	Cumulative cost	Depth	Cumulative cost	-	Depth	Cumulative cost
3	12	1.44×10^7	75	1.49×10^7	390	1.69×10^7	-	12	5.76×10^7
4	16	2.56×10^7	95	2.62×10^7	490	2.88×10^7	-	16	10.24×10^7
5	20	4.0×10^7	115	4.07×10^7	590	4.38×10^7	-	20	16.0×10^7

Table 11.1.: *(above)* Energy approximation ratio for the XXZ model of Eq. (11.7) with $L = 10$ qubits, and $\Delta = 0.5$. The estimates with their uncertainties were calculated using the median and the median absolute deviation of a sample of results obtained by repeating the execution fifty times with different initial conditions. *(below)* Circuit depth expressed as number of CZ gates per qubit, alongside with cumulative number of CZ gates used to reach $\tilde{E}_0$ (See App. D.5). Warm-start VQE approximations (500 epochs of training) are presented alongside VQExDBQA results, executed considering both compiled (GCI) and theoretical (DBI) approaches. For DBI, we compute $\hat{R}_k$ through dense matrix representation so there is no gate count. Longer VQE training (2000 epochs) is reported in the last column of the table.

11.5. Numerical results for XXZ

We study the one-dimensional XXZ Heisenberg model with periodic boundary conditions:

$$\hat{H}_{\mathrm{XXZ}} = \sum_{i=1}^{L} (\hat{X}_i \hat{X}_{i+1} + \hat{Y}_i \hat{Y}_{i+1} + \Delta \hat{Z}_i \hat{Z}_{i+1}),, \tag{11.7}$$

where the subscript denotes the qubit on which the Pauli operator acts. This model is well-characterized by methods including the Bethe ansatz [117, 118, 119], tensor networks [120], and VQE [121].

After training the VQE circuit to a target epoch, we use it to initialize GCI. For each GCI step, we optimize the classically parametrized $\hat{D}_k$ operators (modeled as Ising Hamiltonians; see App. D.5). This parametrization allows $e^{-it\hat{D}_k}$ to be compiled with at most two layers of CZ gates. We compile $e^{-it\hat{H}_{\mathrm{XXZ}}}$ using a second-order Trotter-Suzuki decomposition [97] for short-depth circuits (App. D.6). A full description of our methodology is provided in App. D.5.

Figure 11.3 illustrates the advantages of VQE*x*DBQAfor $L = 10$ qubits. DBQA halves the energy residue when applied to early training epochs and essentially converges to the ground state when applied later. For instance, using a 3-layer ansatz (Fig. 11.2) as a warm start (depth: 12 CZ gates/qubit) followed by one GCI step yields a total VQE*x*DBQA circuit depth of 75 CZ gates/qubit. We quantify performance using the energy approximation ratio:

$$\Delta E := (\tilde{E}_0 - E_0)/E_0 \ , \tag{11.8}$$

where $\tilde{E}_0$ is the energy found by VQE*x*DBQA and E_0 is the true ground state energy.

Figure 11.3 demonstrates an order-of-magnitude improvement, reducing ΔE from $\approx 1\%$ to $\approx 0.1\%$. This improvement is statistically significant, as confirmed by Tab. 11.1, which aggregates results over 50 executions per VQE configuration. DBQA consistently enhances energy estimation across all circuit depths analyzed. Using the method in App. D.5, these energy gains can be translated into fidelity lower bounds [122]. As shown in Tab. 11.2, a single VQE*x*DBQA step achieved a fidelity of $\mathcal{F} \geq 99.6 \pm 0.3\%$.

By increasing the circuit depth of the chosen ansatz of Fig. 11.2 to 20 CZ gates per qubit, it can be trained to reach $\Delta E \approx 0.4\%$. In general, whenever an initialization method is improved to match the previously top performing VQE*x*DBQA circuit, DBQA can be interfaced with that enhanced initialization $\hat{Q}$. Indeed, when initialized with the 20 CZ gates per qubit circuit depth with $\Delta E \approx 0.4\%$, just one DBQA step again gives an order of magnitude gain $\Delta E \approx 0.03\%$. We were not able to reach $\Delta E \approx 0.03\%$ by training deeper VQE circuits.

The VQE*x*DBQA circuits in Fig. 11.3 use a 3-layer VQE warm-start (12 CZ gates/qubit), followed by one GCI step (75 CZ gates/qubit) or two steps

Layers	Warm-start	1 GCI step	2 GCI steps	3 GCI steps
3	0.83 ± 0.06	0.95 ± 0.01	0.993 ± 0.006	0.997 ± 0.003
4	0.89 ± 0.05	0.992 ± 0.007	0.997 ± 0.003	0.998 ± 0.001
5	0.93 ± 0.04	0.996 ± 0.003	0.998 ± 0.002	0.9992 ± 0.0008

Table 11.2.: Fidelity lower bounds [122] (see App. D.5) extending results presented in Tab. 11.1.

(390 CZ gates/qubit). The right panel quantifies the classical optimization runtime via the cumulative number of CZ gates (see App. D.5). Training cost is dominated by VQE for few DBQA steps. For high-fidelity tasks, VQE and DBQA should be used sequentially, as either method alone would require a larger resource expenditure. This cost advantage is confirmed in Table 11.1 for circuits compiled via the GCI formalism (App. D.6). The table also shows that DBI unitaries perform similarly to GCI (next-to-last column), demonstrating the utility of Eq. (11.4).

Additional studies are provided in Appendices D.3, D.4, and D.8. These investigate, respectively: *i)* the scaling for 1D XXZ models of sizes $L \in 4, 6, 8, 10, 12$; *ii)* a non-symmetry-preserving VQE ansatz that requires longer training but achieves a better warm-start energy at shorter circuit depth; and *iii)* target Hamiltonians with next-nearest-neighbor interactions. In all cases, we confirm that the VQExDBQA approach retains its advantage.

11.6. Results on quantum hardware

Finally, we perform a proof of concept experiment on an IBM superconducting quantum chip for the 10 spin XXZ model with open boundary conditions. The initial state is obtained by training two layers of nearest-neighbors Hamming-weight-preserving gates on a classical simulator, acting of the alternating product state $|01\cdots01\rangle$. The diagonal operator consists of single RZ rotations as well as RZZ interactions between next-neighbors, and the time evolution is implemented with a single step of a second order Trotter formula.

Error mitigation is a crucial technique for leveraging the computational power of noisy quantum devices. Our protocol relies on a noise renormalization method, which has proven effective in prior works [123, 124, 125]. The central idea is to assume a depolarizing noise model, estimate its parameters directly on the quantum device, and then invert its effect during post-processing.

We recall that a depolarizing noise model $\mathcal{N}_p[\cdot]$ characterized by a parameter p, acts on the expectation value of a Pauli observable σ under a quantum state ρ as

$$\mathrm{Tr}\left(\sigma \mathcal{N}_p[\rho]\right) = (1-p)\mathrm{Tr}\left(\sigma\rho\right). \tag{11.9}$$

Provided the parameter p is known, one can correct the noisy expectation values by dividing them by the factor $(1-p)$. Here, we can simply estimate this value by replacing the alternating state with the all zero state $|0\cdots0\rangle$, which is invariant under the VQE ansatz. The time evolution is mitigated by performing the first half of the product formula in the forward direction and the second one backward. This method is only effective under depolarizing noise, which is not always an accurate representation of real devices. However, the situation can be improved using Pauli twirling [126]. Readout error are mitigated by calibrating the device [127], as well as twirling the measurement channel [128].

We provide the results, with and without the GCI step, in Table 11.3, which are obtained using 50 twirls with 1000 shot each on the 156 qubits chip ibm_fez. We observe that error mitigation is particularly important when performing the GCI step, as we are able to outperform the bare VQE.

Table 11.3.: Energy computed on the superconducting quantum chip `ibm_fez` with one GCI step on top of a two-layer warm start. Error bars correspond to a 95% confidence interval on the mean computed via Bayesian data augmentation [125]. The target energy is -14.36.

	Warm-start	**1 GCI step**
CZ depth	4	22
statevector	-14.01	-14.27
raw	-13.14±0.01	-10.60±0.01
mitigated	-13.95±0.01	-14.16±0.04

11.7. Outlook

Current quantum hardware can execute circuits with dozens to hundreds of CZ gates per qubit. However, this capacity remains underutilized for high-fidelity ground state preparation due to limitations in existing compilation methods: variational quantum eigensolvers (VQEs) face trainability barriers in large architectures, while quantum phase estimation (QPE) requires fault-tolerant capabilities. We propose that the recently developed double-bracket quantum algorithms (DBQA) approach could provide a means to compile circuits that effectively leverage the available capacity of near-term quantum devices.

The observed synergy between VQE and DBQA, achieving lower energies with shorter training times than VQE alone, is particularly advantageous in the emerging paradigm of cloud-based quantum computing, where circuit execution incurs a token cost. Our implementation [129] within the `Qibo` framework [130] compiles circuits compatible with any

QASM-based API, facilitating the experimental demonstration of these advances.

We note that our numerical analysis did not explicitly incorporate shot noise or hardware decoherence. We anticipate that noise would substantially degrade VQE performance due to the barren plateau phenomenon. In contrast, the optimization of DBI parameters, while beneficial, is optional; certain DBIs are analytically guaranteed to converge even with fixed step durations s_k, albeit less efficiently. Consequently, while noise would impact both methods, we expect DBI to exhibit greater robustness. A full investigation of noise effects is reserved for future work.

While our high-fidelity results rely on a sufficiently accurate initialization, a challenge that grows with system size, Figure 11.3 indicates that DBQA provides significant value even for undertrained initializations, markedly reducing the energy after just one or two steps. If greater precision is required, the algorithmic nature of DBQA allows for further energy reduction at the expense of increased circuit depth. Furthermore, DBQA is agnostic to the initialization method and can be integrated with a wide range of state preparation strategies, as summarized in App. D.9.

We anticipate that DBQAs will maintain their utility in the fault-tolerant era, for instance, as an efficient initialization step for QPE. Techniques like quantum dynamic programming [131] could further reduce the circuit depth of DBQAs, creating a potential sequence: VQE $\rightarrow$ DBQA $\rightarrow$ Depth Reduction $\rightarrow$ QPE. Thus, with respect to circuit depth, DBQAs could serve as a crucial tool to bridge the computational methods of the near-term and fault-tolerant eras.

Finally, a promising direction for future research involves inverting the strategy presented here: using DBQA as a warm-start for variational routines rather than vice versa. This could be explored within a rich landscape of optimization techniques, including Riemannian gradient flows [132], natural gradients [133], and quantum imaginary time evolution [88], with the goal of leveraging their complementary strengths.

All results presented in this chapter are reproducible using the open-source code provided in Ref. [129].

Conclusions

This thesis presents the deployment of the Qibo ecosystem as a full-stack, open-source framework for quantum computing. It focuses specifically on the libraries essential for executing quantum experiments on superconducting chips: Qibolab and Qibocal.

We have described the current state of the Qibolab project, highlighting its major features. The software's abstractions and supported drivers now enable cross-platform instrument benchmarking via arbitrary pulse control, as well as physics experiments based on quantum circuit representations.

For future releases of Qibolab, we plan to extend its capabilities by interfacing with new drivers from commercial and open-source control system vendors and by improving the existing ones to enable new features, such as active reset. Thanks to the library's design, its API can be adapted and scaled for new electronics, including large-scale systems for real-time acquisition and error correction.

In this thesis, we have focused on superconducting chips due to their availability in our affiliated institution's labs. However, we plan to extend Qibolab to support other quantum technologies, such as trapped ions, neutral atoms, and photonics.

Finally, we believe that with the inclusion of Qibolab, Qibo has grown into a powerful tool for the quantum computing community, reducing the software development effort for researchers working on simulation, hardware calibration, and operation.

The release of Qibocal enhances the usability of Qibo as a quantum middleware framework by providing specialized tools for calibrating self-hosted quantum devices. This is achieved through seamless integration with the Qibolab platform and its instrument configuration. Qibocal aims to promote the software standardization of calibration protocols, thereby reducing code redundancy across research groups and laboratories that operate self-hosted quantum hardware.

For future releases, we plan to extend Qibocal's capabilities by defining custom and more efficient calibration protocols to improve gate fidelities, with a primary focus on two-qubit gates and readout. While the current version includes calibration routines for two-qubit gates such as CZ and iSWAP, we are developing new experiments to support architectures with CNOT as a native gate, including those based on cross-resonance interactions and tunable couplers.

Other long-term planned updates for Qibocal aim to improve the API, especially its functionality within Python scripts, and to enhance the handling of reports produced following experiments. At present, manag-

ing reports is up to the user, which frequently turns into a burden that complicates the calibration process. To tackle this issue, we intend to implement a database that can automatically log all experiments and monitor any alterations made to the Platform parameters.

During this time of intense development for Qibolab and Qibocal, the fundamental Qibo framework has been consistently improved. Additional features have been introduced, comprising PyTorch and Clifford backends, along with the Qiboml library for the integration of quantum and hybrid classical-quantum models.

Ultimately, we showcased a ground state preparation application developed with Qibo, concentrating on the newly introduced category of double-bracket quantum algorithms (DBQAs) for Hamiltonian diagonalization. A major difficulty with this approach is that the resulting circuit depth increases exponentially as the number of diagonalization steps rises, rendering it unfeasible for current quantum technologies. We showed that integrating a DBQA with an auxiliary algorithm that can produce a sufficiently precise initial state requires only a limited number of DBQA steps to accomplish high-fidelity ground state preparation. This combined method leads to a total count of two-qubit gates that is manageable for upcoming devices. Numerical simulations were used to validate the method for the Heisenberg model.

Acknowledgments

This thesis and the journey that led to its completion would not have been possible without my supervisor, Stefano Carrazza. I thank him for the trust he placed in me and for the great autonomy he granted, which allowed me to experiment and, most importantly, to make mistakes. A warm thank you goes to all the people who have been involved in the Qibo project, in particular Alessandro Candido for the countless lessons on software development and the importance of always asking questions, Andrea Papaluca for introducing me to ricing and for the discussions about Magic, Simone Bordoni for the thousands of activities he organized, among which the desert excursions will remain unforgettable. A special acknowledgement goes to Andrea Pasquale. I thank him immensely as a colleague for all the time spent initially introducing me to the world of quantum computing and for the countless discussions we had afterwards, from which I always learned something. I thank him as a friend for the support he gave me throughout my journey and the free time we spent together. A special thank you goes to my PhD colleague Matteo Robbiati. Despite the distance, I have always appreciated the good, frank relationship that developed over the years, and I cannot hide the enormous help you gave me in overcoming all the bureaucratic obstacles of this journey. I wish you all the best for your new adventure in Sweden. I thank the Technology Innovation Institute in Abu Dhabi for hosting me during these years of my PhD. A special thank you goes to Frederico Brito for taking the lead of the quantum lab and for the daily work he is carrying out to improve it. Among all the people I met at the center, I extend my heartfelt thanks to Juan Cereijo for his very direct teachings and the pleasant time spent together outside of work. I extend my gratitude to Hayk Sargsyan for our productive exchanges, characterized by his firm logic, and to Luca Ben Hermann for his numerous insightful critiques. I extend my gratitude to Rodolfo Carobene, Marco Gobbo, and the entire Bicocca Quantum Technology group for allowing me to bother them whenever I was in Milan.

Gaja, dovrei scrivere un’altra tesi per elencare i motivi per cui ringraziarti, grazie per avermi fatto sentire meno lontano e per avermi supportato in tutte le scelte che ho fatto, anche se questo ha significato dover rinunciare a qualcosa. Grazie a Ita, Francesco e Alessandro per avermi fatto sentire parte della vostra famiglia. Un ringraziamento finale va a tutta la mia famiglia per tutto l’amore che mi ha dimostrano a distanza e per il costante supporto in tutti questi anni e in quelli futuri.

Non potro’ mai ringraziarvi abbastanza per tutte le cure che avete dato a tutti i miei nonni, in particolare a mio nonno Aldo, un uomo di poche parole ma infinita pazienza.

List of publications

Refereed publications

1. Stavros Efthymiou et al. *Qibolab: an open-source hybrid quantum operating system*. In: *Quantum* 8 (Feb. 2024), p. 1247. ISSN: 2521-327X. DOI:10.22331/q-2024-02-12-1247. URL: http://dx.doi.org/10.22331/q-2024-02-12-1247.

Publications under review

1. Andrea Pasquale, Edoardo Pedicillo et al. *Qibocal: an open-source framework for calibration of self-hosted quantum devices*. Under review. 2024. arXiv: 2410.00101 [quant-ph]. URL: https://arxiv.org/abs/2410.00101.

2. Matteo Robbiati, Edoardo Pedicillo et al. *Double-bracket quantum algorithms for high-fidelity ground state preparation*. Under review. 2024. arXiv: 2408.03987 [quant-ph]. URL: https://arxiv.org/abs/2408.03987.

Publications in conference proceedings

1. Andrea Pasquale et al. *Beyond full statevector simulation with Qibo*. 2024. arXiv: 2408.00384 [quant-ph]. URL: https://arxiv.org/abs/2408.00384.

2. Edoardo Pedicillo, Andrea Pasquale, and Stefano Carrazza. *Benchmarking machine learning models for quantum state classification*. 2023. arXiv: 2309.07679 [quant-ph]. URL: https://arxiv.org/abs/2309.07679.

3. Edoardo Pedicillo et al. *An open-source framework for quantum hardware control*. 2024. arXiv: 2407.21737 [quant-ph]. URL: https://arxiv.org/abs/2407.21737.

4. Li Xiaoyue, Matteo Robbiati, Andrea Pasquale, Edoardo Pedicillo et al. *Strategies for optimizing double-bracket quantum algorithms*. 2024. arXiv: 2408.07431 [quant-ph]. URL: https://arxiv.org/abs/2408.07431.

Publications in preparation

1. Matteo Robbiati, Andrea Papaluca, Andrea Pasquale, Edoardo Pedicillo et al. (2025). *`Qiboml`: towards the orchestration of quantum-classical machine learning*.

A. Qibolab

A.1. Supported drivers

The `Qibolab` package, in its `0.1.0` version released on 11 August 2023, offers comprehensive support for a diverse range of quantum hardware control devices. This includes instruments from Qblox [134], Quantum Machines [135], and Zurich Instruments [136], in addition to RFSoC (Radio Frequency System on Chip) FPGAs (Field Programmable Gate Arrays) which are supported via the `Qick` project [137] and `Qibosoq` [138]. Due to the unique specifications and operating procedures of each device, meticulous implementation is required to deliver seamless control through a unified interface. Specifically:

Qblox The Qibolab tests were performed on a Qblox Instruments cluster [139], a system that treats multiple modular devices as a single, unified instrument. The Qblox Cluster is a scalable 19" rack-mountable unit configurable with up to 20 individual modules, capable of controlling and reading out qubits across a wide frequency range (up to 18.5 GHz). Our specific setup for controlling five superconducting flux-tunable qubits without coupler-mediated interactions consisted of the following modules:

- **QRM-RF** two Qubit Readout Modules [140], each featuring one radio-frequency input channel, one radio-frequency output channel, and two digital markers. These modules enable direct qubit readout for signals between 2–18.5 GHz without the need for external frequency conversion.
- **QCM-RF** three Qubit Control Modules [141], each providing two independent drive channels. Using parametrized pulses, these modules were used to control the five qubits.
- **QCM** two base Qubit Control Modules [142] dedicated to supplying the DC voltages and flux pulses for the qubits' flux channels, which are necessary for implementing two-qubit gates. These modules feature digital-to-analog converters (DACs) with a dynamic output range of 5 Vpp and a sampling rate of 1 Gsps.

Synchronization across all modules within the cluster is managed internally by the Qblox system using its SYNQ [143] protocol. The high-level device control is handled through the Qblox Instruments [36] and Qcodes [144] Python libraries, while low-level communication with the module sequencers is achieved using Q1ASM assembly

code [145]. This configuration provides the capability to control five or more flux-tunable superconducting qubits.

Quantum Machines `Qibolab` has also been validated using a cluster of nine OPX+ controllers [146]. The controllers are interconnected with an all-to-all communication network to facilitate fast feedback operations between any pair of units. System-wide synchronization and clock distribution are managed by Quantum Machines' OPT devices.

Each OPX+ controller in this cluster provides ten analog output channels, ten digital output channels, and two input channels, giving the overall system the capacity to control more than 25 flux-tunable, capacitively coupled qubits. A notable limitation of the OPX+ controllers in our setup, compared to other instruments used in this work, is that they do not handle IQ mixing and upconversion internally. Furthermore, they have a constrained intermediate frequency bandwidth (400 MHz) and a limited output voltage range (0.5 V). These limitations necessitate the use of additional external equipment, such as local oscillators, mixers, and sometimes amplifiers, to properly drive flux qubits.

The `Qibolab` driver manages the entire cluster as a single instrument through the QUA library [35]. This library provides a Python interface for a comprehensive set of low-level operations, which extends beyond basic pulse scheduling to include features like conditional logic, loops, and complex mathematical operations.

Zurich Instruments `Qibolab` was also tested on a Zurich Instruments setup, which integrates several modular devices into a single, synchronized control system. The cluster configuration was as follows:

SHFQC a single SHFQC Quantum Controller [147], capable of controlling the drive and readout for up to 6 superconducting qubits connected to a common readout resonator. It handles IQ mixing and up/downconversion internally using a proprietary scheme [148], providing an instantaneous bandwidth of approximately 1.2 GHz without requiring calibration. The unit also offers an output voltage of 2 Vpp.

HDAWG two HDAWGs [149], each providing up to 8 DC-coupled, single-ended analog output channels. These were used to generate the flux pulses for controlling the qubits and tunable couplers, with an output voltage of up to 5 Vpp.

PQSC a single PQSC (Programmable Quantum System Controller) [150], which synchronizes the other instruments via Zurich Instruments' low-latency ZSync link. The PQSC features 18 ZSync ports to distribute the system clock and synchronize all devices. These links also provide a bidirectional data interface, enabling the PQSC to centrally process qubit readout

results and send trigger signals to the slave instruments for real-time feedback.

The high-level control for this setup is provided by the Python-based LabOneQ library [34]. This configuration allows for the control of five or more flux-tunable superconducting qubits with tunable coupler-mediated interactions.

RFSoCs The RFSoC platforms currently supported by `Qibolab` include the Xilinx RFSoC4x2 [151], ZCU111 [152], and ZCU216 [153]. A key advantage of these FPGAs is their integrated direct RF synthesis, capable of generating signals up to approximately 9.8 GHz. This capability significantly simplifies the experimental setup by removing the requirement for external local oscillators and IQ mixers.

The driver interfaces with the open-source `Qick` firmware via an on-board server called `Qibosoq`. The open-source nature of both the firmware and server helps minimize the costs associated with establishing a new laboratory.

It is important to note, however, that these boards have certain limitations. They support a smaller number of qubits per board compared to other commercial systems, can present synchronization challenges in multi-board configurations, and the supporting software currently offers a more limited feature set.

Beyond the core instruments responsible for pulse synthesis and signal acquisition, a full-featured quantum control platform requires supplementary hardware. Local oscillators are a critical example, used in our setup for frequency up/down-conversion of microwave signals and for pumping TWPAs. Integrating these devices into the control framework is essential, as they require calibration and must be precisely enabled or disabled during operation. To this end, `Qibolab` provides seamless integration of such equipment and, in its `0.1.0` version, includes dedicated drivers for *Erasynth* and *Rohde&Schwarz* local oscillators.

A summary of the supported instruments is provided in Table A.1, with Table A.2 outlining the principal features supported by the drivers in `Qibolab` version `0.1.0`. It is important to note that while certain limitations and missing features exist, these are not necessarily inherent to the hardware and are planned for implementation in future releases of `Qibolab`.

The following descriptions detail the features listed in Table A.2.

Arbitrary pulse sequences The ability to execute any user-defined sequence of pulses, a fundamental requirement for all drivers. This is related to the order and timing of pulses, not the shape of their waveforms.

Arbitrary waveforms The ability to execute pulses with custom, user-defined waveform shapes. Drivers lacking this feature are typically limited to predefined shapes like rectangular, Gaussian, or DRAG.

Device	Firmware	Software
Qblox	0.4.0	qblox-instruments 0.9.0
QM	QOP213	qm-qua 1.1.1
Zurich	Latest (July 2023)[1]	LabOneQ 2.7.0
RFSoCs	Qick 0.2.135	Qibosoq 0.0.3
Erasynth++	-	-
R&S SGS100A	-	QCoDeS 0.37.0

Table A.1.: Outline of the supported devices, along with firmware/software version currently supported.

Multiplexed readout The transmission and acquisition of multiple, spectrally distinct pulses through a single shared transmission line. This is particularly useful for multi-qubit architectures with a common readout resonator.

Hardware classification The capability to perform single-shot qubit state classification (e.g., $|0\rangle$ or $|1\rangle$) in real-time during the execution of a pulse sequence.

Fast reset Allows for the active reset of a qubit to the ground state immediately following a measurement. This feature requires hardware classification to function and enables faster execution of repeated experiments.

Device simulation The ability to simulate the execution of pulse sequences offline, without requiring access to or interaction with the quantum hardware.

RTS frequency Real-Time Sweeping (RTS) of a pulse's frequency, allowing a sequence to be rapidly repeated with different frequency values without recompiling, thus speeding up characterization.

RTS amplitude Real-Time Sweeping of a pulse's amplitude.

RTS duration Real-Time Sweeping of a pulse's duration.

RTS start Real-Time Sweeping of a pulse's start time.

RTS relative phase Real-Time Sweeping of the relative phase between pulses.

RTS 2D The ability to perform two-dimensional real-time sweeps by combining two independent RTS parameters.

Feature	RFSoCs	Qblox	QM	Zhinst
Arbitrary pulse sequences	✓	✓	✓	✓
Arbitrary waveforms	✓	✓	✓	✓[2]
Multiplexed readout	✓	✓	✓	✓
Hardware classification	✗	✓	✓	✓
Fast reset	💻	💻	💻	💻
Device simulation	✗	✗	✓	💻
RTS frequency	✓[3]	✓	✓	✓
RTS amplitude	✓	✓	✓	✓
RTS duration	✗	✓	✓	✓
RTS start	✓	✓	✓	✓
RTS relative phase	✓	✓	✓	✓
RTS 2D any combination	✓	✓	✓	✓
Sequence unrolling	💻	💻	💻	💻
Hardware averaging	✓	✓	✓	✓
Singleshot (No Averaging)	✓	✓	✓	✓
Integrated acquisition	✓	✓	✓	✓
Classified acquisition	✓	✓	✓	✓
Raw waveform acquisition	✓	✓	✓	✓

Table A.2.: Features or limitations of the main drivers supported by `Qibolab 0.1.0`. The features denoted by " ✓ " are supported, " ✗ " means not supported and " 💻 " under development.

Sequence unrolling The process of compiling several iterative subsequences into a single, longer sequence (similar to loop unrolling in computing). This reduces the total number of compilation and communication steps, improving efficiency.

Hardware averaging The device's ability to repeat an experiment multiple times and return a single, hardware-averaged result, reducing data transfer overhead.

Singleshot (No Averaging) The ability to retrieve all individual, non-averaged measurement results from the device for single-shot analysis.

Integrated acquisition The process of demodulating the "in-phase" (I) and "quadrature" (Q) components of a signal and integrating them over the measurement period to produce a complex result [154].

Classified acquisition The capability to perform qubit state classification (0 or 1) on the results of an integrated acquisition.

Raw waveform acquisition The ability to acquire and return the full, non-integrated I and Q waveform data from the acquisition channel.

A.2. Zurich Instruments firmware

Table A.3 shows the firmware version of each Zurich Instruments device used in this work.

Device	Firmware	HDL
HDAWG Control	69121	69120
HDAWG Processing	69121	69080
PQSC	69076	69076
SHFQC	69120	69098

Table A.3.: Zurich FPGA internal controller software and HDL revision.

A.3. Cross-platform benchmark

In this section we provide some more details on the experiments performed for the performance benchmark presented in Sec. 8.4. A more detailed description of these routines is given by [29, 155, 156]. All these experiments were repeated for 4096 shots. For spectroscopies, a relaxation time of 5 μs was used, while for the other experiments it was

set at 300 μs. Relaxation time is the waiting time between consecutive shots to let the qubit relax back to the ground state before the next shot is started.

Resonator spectroscopy consists of a single-tone spectroscopy where a pulse is sent through the readout line and acquired through the feedback line. The frequency of the pulse is swept in a specific range, in our case probing 20 or 100 different frequencies. In the calibration of a 3D (2D) resonator, the amplitudes acquired present a positive (negative) peak at the resonance frequency of the resonator.

Qubit spectroscopy consists of a two-tone spectroscopy where a first pulse is sent to the drive line and a measurement (a readout pulse and an acquisition) is performed right after. The frequency of the drive pulse is swept in a specific range. In the example used for the benchmark, 300 frequencies were analyzed. As per the resonator spectroscopy, the amplitude acquired presents a peak for a specific frequency that, in this case, will be used as the drive pulse frequency.

Rabi amplitude first a drive pulse, at the frequency identified with qubit spectroscopy, is sent through the drive line and a measurement is performed right after. The amplitude of the first pulse is swept in a range composed of, in this case, 75 points. This experiment is used to calibrate the amplitude of the pi-pulse (Pauli-X gate) which rotates the qubits from the $|0\rangle$ state to $|1\rangle$.

Ramsey detuned a first pulse is sent through the drive line. Then, after a delay, a new drive pulse is sent with a delay dependent phase and finally a measurement is performed. The delay between the two drive pulses, and therefore the phase, are swept. This experiment is used to fine tune the drive pulse frequency.

T1 experiment the qubit is excited using a calibrated pi-pulse, then measured after a variable time. The characteristic decay shown by this experiment is used to measure the relaxation time T1 of the qubit.

T2 experiment this experiment is almost identical to the Ramsey detuned experiment, but no additional phase is introduced in the second drive pulse. This enables to compute the characteristic dephasing time T2.

Single shot classification The qubit is first just measured at the initial $|0\rangle$ state, and then excited and measured in the $|1\rangle$ state. The results are used to calibrate the classification between measured states.

Standard RB First, a certain number (iterations) of circuits composed of Clifford gates is randomly generated. These circuits are executed and an average fidelity is computed. Then, new circuits are generated with increased depth and the procedure is repeated. The fidelity is supposed to decrease exponentially with the number of gates per circuit, leading to an estimation of the average error per gate.

B. Qibocal

B.1. Qibocal in action

B.1.1. Coherence at different bias points

As an example of how `Qibocal` can be used to implement custom experiments combining multiple protocols, we compute the values of T_1, T_2^*, and the readout fidelity of a qubit biased at different flux points. Although such a study could be performed in a simpler manner [157], here we deliberately recalibrate the qubit at each flux point in order to showcase the flexibility of the library.

Before running the experiment, we determine how the qubit frequency depends on the applied flux using qubit flux spectroscopy. The measured spectrum is fit with the standard approximation commonly used for these devices (see Eq. 1 in [158], for example).

At each flux point, the calibration procedure consists of:

1. updating the qubit frequency using the fit approximation;
2. running a Rabi experiment to recalibrate the drive pulse amplitude;
3. performing single-shot classification to enable single-shot readout mode;
4. running a Ramsey experiment to fine-tune the drive frequency;
5. repeating single-shot classification to refine the readout;
6. measuring T_1, T_2^*, and the readout fidelity.

The experiment was conducted on a QuantWare qubit controlled by a Quantum Machines [135] cluster running `Qibolab`. As shown in Fig. B.1, the qubit's coherence time T_1 reaches approximately 10 μs at the sweet spot and up to 20 μs at around 150 MHz. Both the echo decay time T_2^* and the readout fidelity are also maximized at the sweet spot, with T_2^* decreasing as expected with detuning.

B.1.2. Pulse optimization with randomized benchmarking

Optimizing pulse shapes against high-level performance metrics is a standard method in the control of quantum devices. This represents a simple instance of *quantum optimal control* [160], wherein pulse parameters are varied to maximize gate fidelities obtained from randomized benchmarking experiments [161], which act as the objective function for the optimization [162, 163]. The `Qibocal` package provides a flexible

platform for developing and benchmarking these techniques. To illustrate this capability, we employ a gradient-free Nelder-Mead optimizer to enhance a $\frac{\pi}{2}$-pulse. The optimizer leverages `Qibocal`'s built-in Clifford randomized benchmarking routine and device parameter update mechanisms. The resulting improvement in the randomized benchmarking decay parameter is shown in Fig. B.2, plotted against the number of optimization steps that propose new values for the pulse amplitude and DRAG coefficient.

B.1.3. Re-calibration after changes in flux background

The performance of superconducting qubits degrades with drift in system and environmental parameters, such as the flux background, requiring periodic recalibration. `Qibocal` enables the construction of automated recalibration protocols by combining its standard routines. We illustrate this with a workflow to recalibrate a $\frac{\pi}{2}$-pulse, which consists of the following steps:

1. Fine-tuning the drive frequency via Ramsey spectroscopy.
2. Optimizing readout parameters through single-shot classification.
3. Recalibrating the pulse amplitude with a Rabi experiment.
4. Re-optimizing the readout with a final single-shot classification.

The fidelity of the recalibrated pulse was quantified using Clifford randomized benchmarking. A controlled qubit detuning was induced by applying a known bias to the flux line. Figure B.3 presents a timeline of gate fidelities, demonstrating the recovery achieved by our recalibration protocol after each induced drift.

B.1.4. Monitoring qubit calibration

`Qibocal` integrates with monitoring tools like Grafana [**Grafana**]. For maximum portability, we deployed these tools using Docker containers. A Grafana container was configured with an automated dashboard layout and support for third-party plugins. These monitoring containers can be scheduled to run at regular intervals, collecting metrics such as qubit T_1, T_2^*, and readout fidelity. As shown in Fig. B.4, these values were logged every thirty minutes for a single qubit, with all data persisted in a dedicated PostgreSQL [164] container. This framework supports more complex workflows, including simultaneous multi-qubit monitoring and automated chip recalibration if metrics degrade beyond a set threshold. Furthermore, Docker's modular architecture allows for additional containers to monitor metrics like QPU usage or cryostat temperature.

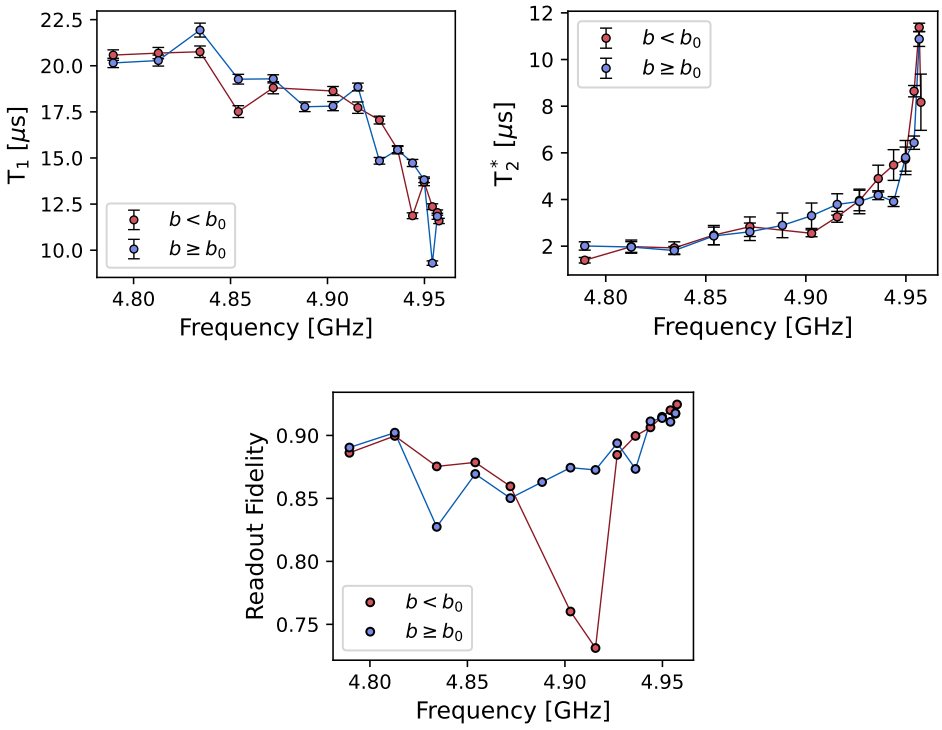


Figure B.1.: Measurements of T_1, T_2^* and readout fidelity as the qubit is operated at different frequencies. The blue curve indicates that the detuning is induced by increasing the flux, while the red curve corresponds to the case where we are decreasing flux. For each point we follow the recalibration procedure described in Sect. B.1.1. The behavior seen for T_1 measurements is consistent with the increase of the qubit's Purcell protection (see [159] for a systematic study about the spontaneous emission of Transmon qubits).

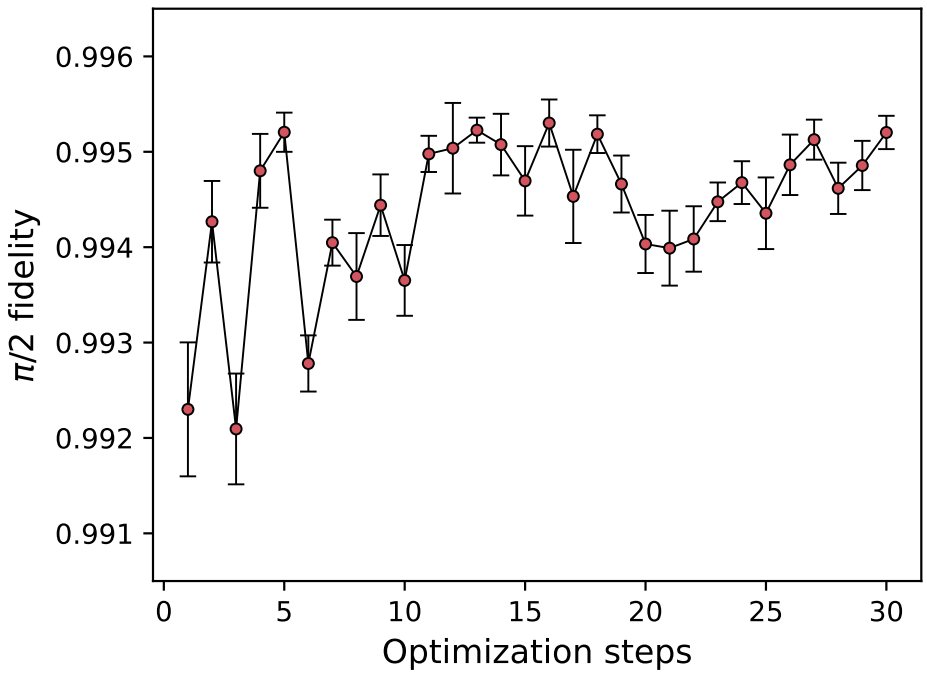


Figure B.2.: The $\frac{\pi}{2}$-pulse fidelity extracted from randomized benchmarking (RB) is used as the objective function of a Nelder-Mead optimization of the $\frac{\pi}{2}$-pulse amplitude and DRAG parameter. The resulting $\frac{\pi}{2}$-pulse fidelity after each step in the optimization is shown. The optimization results in an increase in fidelity with few evaluations of the cost function.

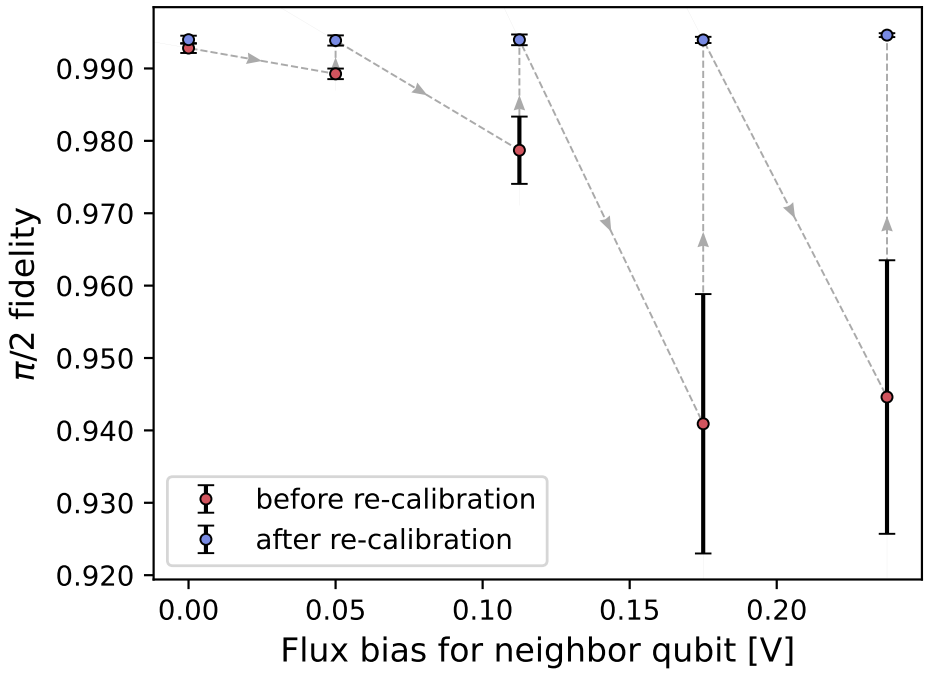


Figure B.3.: Re-calibration of a qubit's $\frac{\pi}{2}$-pulse after (controlled) changes of its flux background. Applied voltage to flux line of neighbouring qubit and randomized benchmarking $\frac{\pi}{2}$-fidelity before (red) and after (blue) re-calibration against time. The gray dashed arrows describes how the calibration changes over time. Flux bias is relative to the qubit's calibrated sweetspot.

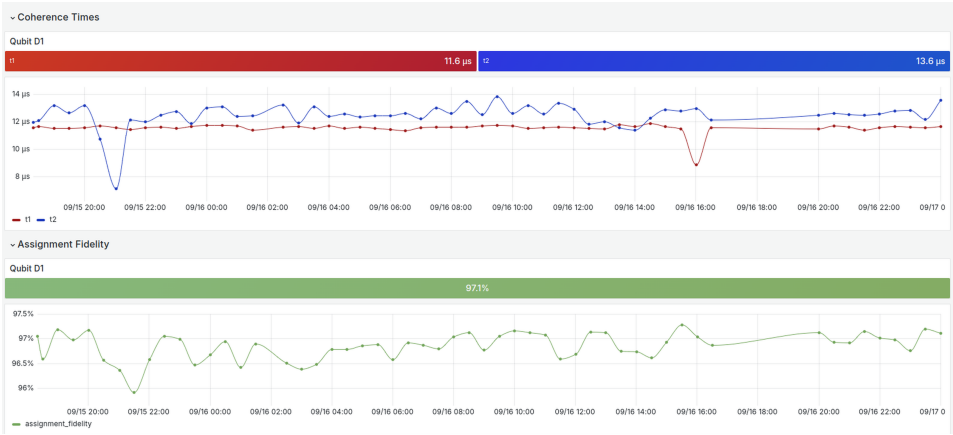


Figure B.4.: Monitoring of coherence times and readout fidelity using `Qibocal`.

C. Standard Randomized Benchmarking

The key idea of randomized benchmarking (RB)[58] is that averaging the error process over the uniform space of unitaries yields an effective depolarizing channel, which maps any pure state to the maximally mixed state. This uniform distribution of unitaries is defined by the **Haar measure**. It can be shown[58] that the average induced error is proportional to the depolarization probability.

However, sampling directly from the Haar measure is inefficient. A practical simplification was proposed in [165] by restricting the set of unitaries to the Clifford group, which consists of unitary operations that map the Pauli operators onto themselves. The Clifford group has several advantages: its size is finite for a fixed Hilbert space dimension, and since it forms a group, the inverse of any sequence of Clifford operations is guaranteed to also be a Clifford operation.

The generic procedure for randomized benchmarking is as follows:

1. Initialize the system in the ground state.
2. For each sequence length m, draw a random sequence of Clifford gates.
3. Compute the inverse gate that ideally returns the system to $|0\rangle$.
4. Apply the sequence followed by the inverse, then perform a measurement.
5. Repeat the procedure for many random sequences of the same length, and for multiple values of m.

This approach works because it has been shown [166] that randomization over the Clifford group again produces an effective depolarizing channel,

$$\rho \longrightarrow \frac{d}{2} I + (1-d)\rho, \tag{C.1}$$

with depolarization probability d. Measuring the survival probability,i.e., the probability of finding the qubit in $|0\rangle$, as a function of sequence length m yields the characteristic decay,

$$F(m) = Ap^m + B, \tag{C.2}$$

where $1-p$ is the depolarization rate, while A and B account for state preparation and measurement (SPAM) errors. From this, the average error per Clifford gate is extracted as

$$r = \frac{(2^n-1)(1-p)}{2^n}, \tag{C.3}$$

where n is the number of qubits. The error per physical gate can then be obtained by dividing the Clifford error by the average number of primitive gates per Clifford, typically 1.875, for single qubit RB.

A key advantage of RB is that it isolates the intrinsic gate error, independently of SPAM errors, which are absorbed into the parameters A and B.

Standard RB provides only coarse-grained information about the errors in a system. Interleaved RB, instead, is designed to estimate the error rate of a specific gate of interest. The procedure is nearly identical to standard RB, with the sole difference being that after each randomly chosen Clifford gate, the target gate C is inserted.

From the exponential decay fit, we extract the depolarization parameter p_C, which allows us to estimate the error rate associated with the gate C as

$$r_C = \frac{(2^n-1)(1-\frac{p_C}{p})}{2^n}, \tag{C.4}$$

where p is the parameter from the standard RB.

D. Supplementary materials on double bracket quantum algorithm

D.1. Summary of double-bracket quantum algorithms

We first summarize the qualitative aspects of our approach with their main references before discussing them step-by-step with explicit formulas. Our method is derived from *i)* double-bracket flows which are non-linear differential equations for matrices [82, 167]. Because implementing these continuous flows exactly on a quantum computer is unfeasible, we instead use *ii)* discrete iterations [81, 83, 84]. The form of these iterations is *iii)* recursive, with each step depending on the iterated matrix rather than the original input. This allows us to "dress" the input Hamiltonian with an initialization unitary without affecting the algorithm's structure. To address the problem that quantum recursion typically forces an exponential increase in either circuit *iv)* depth or *v)* width, we *vi)* extend the durations of the discretization steps to maximize the cost function improvement per step. Furthermore, we *vii)* allow for variational optimization beyond the operators of the continuous flow, again to maximize the *viii)* cost function gain in each iteration.

For completeness, we present *i)* an example of a double-bracket flow, specifically the Brockett flow [168, 82]. Given an input matrix A_0, the flow A_ℓ at any $\ell \in \mathbb{R}$ is the solution to

$$\partial_\ell A_\ell = [[N, A_\ell], A_\ell]. \tag{D.1}$$

Provided the matrix N is diagonal and has a non-degenerate spectrum, the flow converges such that A_∞ is diagonal with its eigenvalues ordered according to those of N for almost all initial matrices A_0 [168, 82].

The Euler scheme discretization of this differential equation, with a step size $\Delta\ell \in \mathbb{R}$, is given by

$$A^{(k+1)} = A^{(k)} + \Delta\ell[[N, A^{(k)}], A^{(k)}]. \tag{D.2}$$

This update rule constitutes the linear expansion of the unitary recursions we will now define.

Let $\hat{H}_0$ be a Hamiltonian and assume that we are given a sequence of hermitian diagonal operators $\hat{D}_0, \hat{D}_1, \ldots$. We define the *ii) double-bracket iteration* (DBI) as the *iii)* recursion starting with $k = 0$ and

$$\hat{H}_{k+1} = e^{s_k \hat{W}_k} \hat{H}_k e^{-s_k \hat{W}_k} , \tag{D.3}$$

where $\hat{W}_k$ is a commutator bracket

$$\hat{W}_k = \hat{D}_k \hat{H}_k - \hat{H}_k \hat{D}_k = [\hat{D}_k, \hat{H}_k] . \tag{D.4}$$

Next, we single out the *double-bracket unitary*

$$\hat{U}_{k+1} = \hat{U}_k e^{-s_k \hat{W}_k} \tag{D.5}$$

which appears in the DBI recursion equation.

As discussed in the following section, the unitary $\hat{U}_k$ can be approximated using group commutator formulas, which reduce the required quantum computation to sequences of standard Hamiltonian simulations. However, as we will demonstrate through explicit gate counting, this approach leads to an exponential growth in the circuit *iv)* depth. An alternative is to employ the quantum dynamic programming [131] method to execute the recursion. This technique requires only polynomial circuit depth, but achieves this at the cost of an exponential increase in the circuit *v)* width.

We will say that

$$\hat{H}_k(s) = e^{s \hat{W}_k} \hat{H}_k(0) e^{-s \hat{W}_k} \tag{D.6}$$

is a *double-bracket rotation* (DBR) since it satisfies a Heisenberg equation involving two, not one, brackets

$$\partial_s \hat{H}_k(s) = [[\hat{D}_k, \hat{H}_k(0)], \hat{H}_k(s)] . \tag{D.7}$$

This allows us to define a greedy optimization scheme of a cost function $f : \mathbb{C}^{2^L \times 2^L} \to \mathbb{R}_{\geq}$ by considering *vi)* a global minimum of the cost function

$$s_k = \mathrm{argmin}_{s \in \mathbb{R}} f(\hat{H}_k(s)) . \tag{D.8}$$

Given this optimizer conditioned implicitly on the $\hat{D}_k$ operator we can *vii)* optimize these operators e.g. by repeatedly taking a starting guess, updating it and finding the minimum of f. We then set $\hat{H}_{k+1} = \hat{H}_k(s_k)$.

Finally, we discuss *viii)* the explicit form of the cost functions. For the DBR ansatz, the cost function is defined by the magnitude of the off-diagonal elements of $\hat{H}_k$. Specifically, we let $\sigma(\hat{H}_k)$ denote the off-diagonal restriction of $\hat{H}_k$; that is, $\sigma(\hat{H}_k)$ shares the same off-diagonal matrix elements as $\hat{H}_k$, but its diagonal elements are set to zero.

Any matrix norm applied to $\sigma(\hat{H}_k)$ would quantify the magnitude of the off-diagonals. We choose the Hilbert-Schmidt norm, which originates from the Hilbert-Schmidt scalar product [169]:

$$\langle \hat{A}, \hat{B} \rangle_{\mathrm{HS}} = \mathrm{tr}[\hat{A}^\dagger \hat{B}]. \tag{D.9}$$

The squared norm is then given by $|\hat{A}|^2_{\mathrm{HS}} = \mathrm{tr}[\hat{A}^\dagger \hat{A}]$. For this choice, the Taylor expansion of the norm difference takes the form:

$$|\sigma(\hat{H}k)|^2_{\mathrm{HS}} - |\sigma(\hat{H}k-1)|^2_{\mathrm{HS}} = -2sk-1\langle \hat{W}_{k-1}, [\hat{H}_{k-1}, \sigma(\hat{H}_{k-1})] \rangle_{\mathrm{HS}} + O(s^2_{k-1}). \tag{D.10}$$

This expression implies that, provided the sign of $\hat{D}_k$ is chosen correctly, the Hilbert-Schmidt norm of the off-diagonal part will decrease. Through iteration, the process converges towards a fixed point where the matrix is diagonal. Since this diagonalization is effected by the unitaries $\hat{U}_k$, applying them to computational basis states yields an approximation to an eigenstate.

In this work, we demonstrate that DBIs can reduce the expected energy, defined as

$$E_k = \langle \psi_k | \hat{H}_0 | \psi_k \rangle = \langle 0 | \hat{H}_k | 0 \rangle, \tag{D.11}$$

where

$$|\psi_k\rangle = \hat{U}_k |0\rangle. \tag{D.12}$$

Minimizing the energy cost function $f_E(\psi) = \min \langle \psi | \hat{H}_0 | \psi \rangle$ prepares an eigenstate, though not necessarily a specific one. To target eigenstate preparation more directly, one can use the energy fluctuation,

$$\Xi(\psi) = \sqrt{\langle \psi | \hat{H}_0^2 | \psi \rangle - \langle \psi | \hat{H}_0 | \psi \rangle^2}, \tag{D.13}$$

which is an experimentally measurable cost function. We define the energy fluctuation during a DBI as

$$\Xi_k = \sqrt{\langle \psi_k | \hat{H}_0^2 | \psi_k \rangle - \langle \psi_k | \hat{H}_0 | \psi_k \rangle^2} = \sqrt{\langle 0 | \hat{H}_k^2 | 0 \rangle - \langle 0 | \hat{H}_k | 0 \rangle^2}. \tag{D.14}$$

D.2. Details on group commutator iterations

We adopt the notation

$$\hat{A}_{k+1} = \hat{V}_k^\dagger \hat{A}_k \hat{V}_k \tag{D.15}$$

to describe the evolution under group commutator iterations. Here, the recursion step unitary $\hat{V}_k$ is derived from various approximations to the ideal double-bracket rotation unitary

$$\hat{U}_k = e^{-s_k[\hat{D}_k, \hat{A}_k]}. \tag{D.16}$$

The group commutator unitary for Hermitian generators $\hat{A}$ and $\hat{B}$ is defined as

$$\hat{V}^{(\mathrm{GC})}(\hat{A},\hat{B}) \equiv e^{i\hat{A}}e^{i\hat{B}}e^{-i\hat{A}}e^{-i\hat{B}}. \tag{D.17}$$

For an iterative step, we set

$$\hat{V}_k = \hat{V}^{(\mathrm{GC})}(\sqrt{s_k}\hat{A}_k, -\sqrt{s_k}\hat{D}_k) = e^{-s_k[\hat{D}_k,\hat{A}_k]} + O(s_k^{3/2}), \tag{D.18}$$

which provides an approximation since, in general, $\hat{V}^{(\mathrm{GC})}(\hat{A},\hat{B}) \approx e^{-[A,B]}$.

We consider this specific ordering because, when rotating $\hat{A}_k$ in Eq. (D.15), we can equivalently use the reduced group commutator formula:

$$\hat{V}^{(\mathrm{RGC})}(\hat{A},\hat{B}) = e^{i\hat{B}}e^{-i\hat{A}}e^{-i\hat{B}}. \tag{D.19}$$

If we instead set

$$\hat{V}_k = \hat{V}^{(\mathrm{RGC})}(\sqrt{s_k}\hat{A}_k, -\sqrt{s_k}\hat{D}_k), \tag{D.20}$$

we do not directly approximate the double-bracket unitary $\hat{U}_k$. However, because $e^{-i\hat{A}_k}\hat{A}_k e^{i\hat{A}_k} = \hat{A}_k$ holds, this choice yields the same transformed operator $\hat{A}_{k+1}$ in Eq. (D.15).

Finally, we consider a generalization of the group commutator from Ref. [95] that provides a higher-order approximation. The principle is to approximate the evolution generated by the commutator of two generators with a product of their individual evolutions. The specific formula, given by Eq. (8) in the reference, is:

$$\hat{V}^{(\mathrm{HOPF})}(\hat{A},\hat{B}) = e^{i\phi\hat{A}}e^{i\phi\hat{B}}e^{-i\hat{A}}e^{-i(\phi+1)\hat{B}}e^{i(1-\phi)\hat{A}}e^{i\hat{B}}, \tag{D.21}$$

where $\phi = \frac{1}{2}(\sqrt{5}-1)$. This construction is designed such that $\hat{V}^{(\mathrm{HOPF})}(\hat{A},\hat{B}) \approx e^{-[\hat{A},\hat{B}]}$. To implement this within a group commutator iteration, we set

$$\hat{V}_k = \hat{V}^{(\mathrm{HOPF})}(\sqrt{s_k}\hat{A}_k, -\sqrt{s_k}\hat{D}_k) = e^{-s_k[\hat{D}_k,\hat{A}_k]} + O(s_k^2). \tag{D.22}$$

This construction yields a higher-order approximation by canceling the first- and third-order terms in the expansion, thus preserving the essential second-order term required to approximate the double-bracket rotation unitary. We refer to this as the third-order group commutator to emphasize the cancellation of the third-order term. Notably, the formula involves three exponential applications per generator (e.g., three with $\hat{A}$ and three with $\hat{B}$), whereas the standard group commutator uses only two per generator.

For our numerical simulations, we again employ a reduced form that achieves cancellation. We define:

$$\hat{V}^{(\mathrm{RHOPF})}(\hat{A},\hat{B}) = e^{i\phi\hat{B}}e^{-i\hat{A}}e^{-i(\phi+1)\hat{B}}e^{i(1-\phi)\hat{A}}e^{i\hat{B}}, \tag{D.23}$$

with $\phi = \frac{1}{2}(\sqrt{5}-1)$ as before. We then set the iteration unitary to

$$\hat{V}_k = \hat{V}^{(\mathrm{RHOPF})}(\sqrt{s_k}\hat{A}_k, -\sqrt{s_k}\hat{D}_k) = e^{-s_k[\hat{D}_k,\hat{A}_k]} + O(s_k^2), \tag{D.24}$$

gaining the benefit of requiring one fewer Hamiltonian simulation query per step.

D.2.1. Explicit unfolding for GCI with RGC

As defined previously, we initialize the iterative process with

$$\hat{A}_0 = \hat{U}_{\boldsymbol{\theta}^*}^\dagger \hat{H}_0 \hat{U}_{\boldsymbol{\theta}^*}, \tag{D.25}$$

ensuring the energy expectation value matches the VQE loss function:

$$\langle 0|\hat{A}_0|0\rangle = \langle \psi_0(\boldsymbol{\theta})|\hat{H}_0|\psi_0(\boldsymbol{\theta})\rangle. \tag{D.26}$$

In this section, we see how a Group Commutator Iteration (GCI) for $\hat{A}_k$ is composed only by a unitary of the VQE unitary $\hat{U}_{\boldsymbol{\theta}^*}$, the input Hamiltonian evolutions $e^{-it\hat{H}_0}$, and the diagonal evolutions e^{-itD_k}.

The first iteration step uses $\hat{A} = r_0\hat{H}_0$ and $\hat{B} = -r_0\hat{D}_0$ with $r_0 = \sqrt{s_0}$, yielding the unitary:

$$\hat{V}_1 = e^{-ir_0\hat{D}_0}\hat{U}_{\boldsymbol{\theta}^*}^\dagger e^{-ir_0\hat{H}_0}\hat{U}_{\boldsymbol{\theta}^*} e^{ir_0\hat{D}_0}. \tag{D.27}$$

Consequently, the transformed operator is:

$$\hat{A}_1 = \hat{V}_1^\dagger \hat{U}_{\boldsymbol{\theta}^*}^\dagger \hat{H}_0 \hat{U}_{\boldsymbol{\theta}^*}\hat{V}_1. \tag{D.28}$$

This implies the resulting state's energy expectation is:

$$\langle 0|\hat{A}_1|0\rangle = \langle \psi_1(\boldsymbol{\theta})|\hat{H}_0|\psi_1(\boldsymbol{\theta})\rangle, \tag{D.29}$$

where the state at step one is defined as:

$$|\psi_1(\boldsymbol{\theta})\rangle = \hat{U}_{\boldsymbol{\theta}^*}\hat{V}_1|0\rangle. \tag{D.30}$$

Here, $\hat{V}_1$ acts directly on the computational basis state $|0\rangle$, not on the VQE state. This suggests that the role of the DBQA step is to subtly entangle the computational basis state, preparing it for the subsequent application of the VQE unitary.

However, we note that for a traceless $\hat{D}_0$, the action is trivial:

$$e^{is_0\hat{D}_0}|0\rangle = |0\rangle, \tag{D.31}$$

or otherwise results in an immaterial global phase. Therefore, we can express the state more explicitly as:

$$|\psi_1(\boldsymbol{\theta})\rangle = \hat{U}_{\boldsymbol{\theta}^*} e^{-is_0\hat{D}_0}\hat{U}_{\boldsymbol{\theta}^*}^\dagger e^{-is_0\hat{H}_0}\hat{U}_{\boldsymbol{\theta}^*}|0\rangle. \tag{D.32}$$

This form clarifies that the Hamiltonian evolution $e^{-is_0\hat{H}_0}$ in the first step acts directly on the low-lying VQE state $\hat{U}_{\boldsymbol{\theta}^*}|0\rangle = |\psi(\boldsymbol{\theta})\rangle$.

To perform a second GCI step with parameter $r_1 = \sqrt{s_1}$, we define the unitary:

$$\hat{V}_2 = e^{-ir_1\hat{D}_1}\hat{V}_1^\dagger e^{-ir_1\hat{A}_0}\hat{V}_1 e^{ir_1\hat{D}_1} = e^{-i(r_0\hat{D}_0+r_1\hat{D}_1)}\hat{U}_{\boldsymbol{\theta}^*}^\dagger e^{ir_0\hat{H}_0}\hat{U}_{\boldsymbol{\theta}^*} e^{ir_0\hat{D}_0}\hat{U}_{\boldsymbol{\theta}^*}^\dagger e^{-ir_1\hat{H}_0}\hat{U}_{\boldsymbol{\theta}^*} e^{-ir_0\hat{D}_0}\hat{U}_{\boldsymbol{\theta}^*}^\dagger e^{-ir_0\hat{H}_0}\hat{U}_{\boldsymbol{\theta}^*} e^{i(r_0\hat{D}_0+r_1\hat{D}_1)}. \quad \text{(D.33)}$$

A key observation is that this expression contains 3 queries to the evolution operator $e^{-it\hat{H}_0}$ of the input Hamiltonian.

Using $\hat{V}_2$, the resulting state becomes:

$$|\psi_2(\boldsymbol{\theta})\rangle = \hat{U}_{\boldsymbol{\theta}^*}\hat{V}_1\hat{V}_2|0\rangle. \quad \text{(D.34)}$$

Again, since the final diagonal unitary $e^{i(r_0\hat{D}_0+r_1\hat{D}_1)}$ in $\hat{V}_2$ contributes only a global phase, its implementation can be omitted (note this is not true for the diagonal operations in $\hat{V}_1$). Consequently, two steps of GCI require a total of 4 queries to $e^{-it\hat{H}_0}$.

We refer to Eq. (D.33) as the unfolded form of the second GCI step. This process of recursively substituting the definition of $\hat{V}_k$ is repeated for subsequent steps until every unitary in the resulting sequence is expressed solely in terms of evolutions generated by $\hat{D}_k$ or the original Hamiltonian $\hat{H}_0$.

For the numerical simulations, we employ the third-order higher-order product formula to approximate the double-bracket rotation, rather than the first-order method. The unfolding procedure proceeds analogously for this higher-order formula, albeit involving a larger number of queries to the input Hamiltonian simulation per step.

D.3. Tackling different XXZ sizes

In this section, we analyze one-dimensional XXZ models ranging from 4 to 12 qubits. For each system size, we train a Hamming-weight preserving ansatz with four layers, using the architecture shown in Fig. 11.2. The training employs an Adam optimizer for 300 iterations, with details provided in Appendix D.5. We repeat this training process 10 times for each configuration, initializing the VQE circuit each time with a new set of angles sampled from a uniform distribution $\mathcal{U}(-\pi, \pi)$.

We then apply three steps of preconditioned GCI to each trained VQE model, performing the loading at epochs 100, 200, and 300. Each GCI step is optimized via fifty iterations of a CMA-ES algorithm [116] (further details on the GCI optimization are provided in Appendix D.5).

The obtained results are shown in Tab. D.1, and highlight how the VQE*x*DBQA approach starts to be impactful on the ground state preparation with a number of qubits $L \geq 8$. In fact, the application of GCI after the VQE training allows us to reach a remarkable increasing of the fidelity lower bounds even considering only one GCI step. For number of qubits $L < 8$ the obtained bounds contain so many nines that it is not useful to report them. In those cases, the VQE alone is recommended. Alongside with the fidelity lower bounds, we show the depth of the executed circuits computed as total number of two-qubit gates per qubit. We observe the depth of the circuits remains approximately constant. This is due to how we compile the GCI in terms of quantum gates. In fact, the contributions due to VQE absorption in the compilation are fixed by the number of layers, and the size of the oracles implementing the Hamiltonian and diagonal evolutions are fixed by the number of time steps and the order of the respective Trotter-Suzuki decomposition. Importantly, these results are obtained assuming each of the VQE circuits composed of four layers. Increasing the size of the problem may require larger VQE circuits, or - in general - more expensive warm start approximation unitaries. In those cases, the depth of the required VQE*x*DBQA circuit will suffer of the scalability inherited from the pre-conditioning technique.

D.4. VQE using Hardware-Efficient ansatz

In this section, we evaluate the performance of DBQAxVQE using a hardware-efficient ansatz (HEA) for the XXZ Hamiltonian. In principle, this ansatz is more expressive than the Hamming-weight preserving ansatz shown in Fig. 11.2, as it can potentially cover the entire Hilbert space given sufficient circuit expressivity.

This expressivity implies that the optimal solution resides within the algorithm's search space. However, it also makes the training process more challenging, as it does not incorporate the prior physical knowledge (like particle number conservation) exploited by the Hamming-weight preserving ansatz. Consequently, the optimization must also navigate regions of the Hilbert space that do not contain the target solution.

A practical comparison of the HEA circuit with the ansatz in Fig. 11.2 reveals that the former has more parameters but a lower number of CZ gates. The final choice of ansatz must therefore balance expressivity against hardware-specific constraints, such as qubit connectivity and the calibration accuracy of two-qubit gates.

We follow the methodology from the main text, performing $N = 5$ training instances per configuration. Figure D.1 displays a sample VQE*x*DBQA execution with a fixed random seed. The VQE training exhibits a swift initial decline in the cost function before saturating with minimal further improvement, irrespective of layer count. Conversely, DBQA halves the energy residue when applied early (epochs $1000, 2000$) and nearly converges to the ground state at later stages.

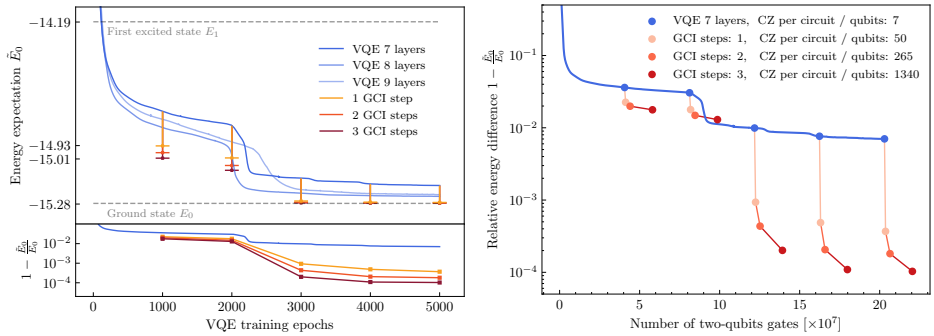


Figure D.1.: One example of VQExDBQA for XXZ using a hardware efficient ansatz and obtained fixing the simulation random seed; the image is intended to provide qualitative information about the impact of VQExDBQA. A more robust study of the performance is presented in Tab. 11.1.*(left)* Training of VQE (blue lines) for 7, 8, and 9 layers (hues of blue) achieve within 500 training epochs ground state energy residue of about 1%. We initialize DBQA with VQE for selected epochs $\in [1000, 2000, 3000, 4000, 5000]$ where we apply a DBQA optimized in its parameters with CMA-ES [116]. *(right)* Token cost estimates of VQExDBQA by counting the total number of two-qubit gates required to execute the complete protocol: training the VQE until a target epoch and then optimizing and applying the DBQA.

The lower left plot indicates that initializing DBQA at the first training plateau places it within the ground state's basin of attraction, enabling exponentially fast convergence.

The right plot displays the cumulative CZ gate count and circuit depth (CZ gates per qubit) for the VQExDBQA protocol. The training cost is dominated by VQE when only a few DBQA steps are used. However, DBQA achieves greater energy optimization with fewer queries. For high-fidelity ground state preparation, the sequential use of both methods is most efficient, as either alone would require excessive resource expenditure. A detailed performance analysis is provided in Tab. D.2 (using the relative difference metric of Eq. (D.35)) and Tab. D.3 (using the fidelity lower bound from Eq. (D.36)).

D.5. Details about numerical simulations

This section details the methodology behind our numerical simulations and the associated computational costs for the VQE and VQExDBQA approaches, as reported in this work's tables.

All simulations were executed using `Qibo` [130, 17, 20, 170, 171, 172,

173], an open-source quantum computing framework extensively employed for quantum machine learning algorithms in both simulation [174, 175, 176, 177, 178, 179, 180] and on quantum hardware [181, 182, 183]. The variational optimizations were conducted via the `Qibo` interface, which leverages established Python frameworks including `keras` [184], `tensorflow` [14], `scipy` [185], and `pycma` [116].

D.5.1. VQE training and computational cost

Training description — We train the VQE using a gradient-based optimization with the Adam optimizer [186], known for its effectiveness with large models. The learning rate was set to 0.05 following a hyperparameter search over a grid from 0.1 to 0.001; all other hyperparameters retain their default Keras values.

We explore two VQE architectures: the Hamming weight preserving ansatz from Fig. 11.2 and a hardware-efficient ansatz built from RY, RZ, and CZ gates. The target Hamiltonians are instances of the general Heisenberg model. The first case, denoted as XXZ, includes nearest-neighbor interactions with a $\hat{Z}$ penalty of $\Delta = 0.5$. The second, more complex case, denoted as J_1-J_2, also incorporates next-nearest-neighbor interactions.

For a given VQE ansatz and target Hamiltonian, we assess its expressibility by scaling the number of layers. Both ansätze were trained with depths ranging from 3 to 9 layers, with each configuration repeated across five distinct initializations. The initial parameters were randomly sampled from a uniform distribution $\mathcal{U}(-\pi, \pi)$.

Based on the initial accuracy in ground state approximation, we selected optimal circuit sizes for more exhaustive error analysis. For the XXZ target, we trained Hamming-weight preserving ansätze of 3, 4, and 5 layers for 2000 epochs, and hardware-efficient ansätze of 7, 8, and 9 layers for 5000 epochs. For the more complex J_1-J_2 target, we trained Hamming-weight preserving ansätze of 3 to 6 layers for 2000 epochs.

For each combination of target Hamiltonian, ansatz, and number of layers, we perform fifty independent training runs. Each run initializes parameters from the uniform distribution $\mathcal{U}(-\pi, \pi)$. The resulting ground state energy approximation $\tilde{E}_0$ is evaluated using the relative difference (RD) from the true ground state energy E_0:

$$\text{RD} = 1 - \frac{\tilde{E}_0}{E_0}. \tag{D.35}$$

Alternatively, we compute a lower bound on the fidelity [122]:

$$\mathcal{F} = 1 - \frac{\tilde{E}_0 - E_0}{E_1 - E_0}, \tag{D.36}$$

where E_1 is the energy of the first excited state.

The final result for each set of fifty runs is summarized by the median of the RD or $\mathcal{F}$ values, denoted median(x), where x is the list of results. The associated uncertainty is quantified using the median absolute deviation (MAD):

$$\text{MAD} = 1.4826 \cdot \text{median}\big(|x_i - \text{median}(x)|\big). \tag{D.37}$$

We employ these robust estimators to mitigate the impact of outliers, which can occur during VQExDBQA execution. When the initial VQE approximation is insufficiently accurate, the DBQA may fail to lower the energy further due to the use of a global cost function. Furthermore, the intense optimization cost of the non-compiled DBI can sometimes prevent the optimizer from converging to an optimal parameter configuration within the training time.

Computational cost — The computational cost of VQE training is determined by the total number of two-qubit gates in the circuit and the number of expectation value evaluations required for predictions and gradients. For gradient-based optimization on quantum hardware, gradients are computed via parameter shift rules [187, 188]. As our circuit is parametrized by rotational gates, the partial derivative with respect to any parameter θ requires two expectation value evaluations. Since we use the Adam optimizer, the full gradient vector must be computed every iteration. The total number of CZ gates is given by:

$$N_{CZ}^{VQE}(e) = k \cdot p \cdot e \cdot n_{CZ}^{VQE}, \tag{D.38}$$

In Eq. (D.38), e is the number of Adam iterations (epochs), p is the number of circuit parameters, and n_{CZ}^{VQE} is the number of CZ gates in the chosen ansatz. The constant k represents the number of expectation values needed per parameter shift. For the hardware-efficient ansatz, $k = 2$. For the Hamming weight preserving ansatz, $k = 4$, as each RBS gate is decomposed into two rotations sharing the same angle θ [111]. This decomposition also dictates that the number of CZ gates in the Hamming-weight preserving ansatz is twice its number of RBS gates.

D.5.2. VQExDBQA procedure and computational cost

VQExDBQA procedure — The algorithm requires an initial ground state approximation, which we obtain from VQE (though any preparation method is suitable). This approximation preconditions the target Hamiltonian, which is then rotated per Eq. (11.2). The subsequent key step is compiling the DBI circuit, following the procedure in Sec. D.6.

To analyze computational cost, we use the VQE trainings from the previous section as preconditioning approximations. We apply the VQExDBQA protocol to a VQE state stopped at epoch e and compute the resulting energy via Eq. (D.35). Following the methodology of Sec. D.5.1, we report the median and median absolute deviation of the RD values from multiple runs.

Computational cost — A complete assessment of the algorithm's computational expense must account for both the initial ground state approximation and the compilation of the GCI into a quantum circuit. The total number of two-qubit gates required is the sum of two contributions. The first is the cost of the VQE pre-training, given by Eq. (D.38). The second is the cost of executing the full VQExDBQA circuit, which involves an exponential repetition of the VQE unitary with increasing DBI steps, augmented by the overhead from the Hamiltonian decomposition detailed in Sec. D.6. We denote the total CZ gates in this final compiled circuit as n_{CZ}^{DBQA}. An additional optimization cost arises from tuning the GCI's parameters. For this work, the diagonal operators $\hat{D}_k$ are parameterized as a classical NN Ising model:

$$\hat{D}_k(B^{(k)}, J^{(k)}) = \sum_{i=0}^{N} (\alpha_i^{(k)} \hat{Z}_i + \beta_i^{(k)} \hat{Z}_{i+1} \hat{Z}_i), \tag{D.39}$$

where N is the number of chain sites. The operator is parametrized by coefficients $\alpha_i i = 0^N$ and $\beta_i i = 0^N$, and the superscript k indicates this optimization is repeated for each DBQA step. We employ Scipy's Powell and CMA-ES optimizers to find an optimal DBI configuration, a process that contributes an additional cost. This final contribution to the total two-qubit gate count is calculated by multiplying n_{CZ}^{DBQA} by the total number of cost function evaluations n_{fval} performed by the optimizers. Thus, the total number of two-qubit gates required for the full VQExDBQA protocol is:

$$N_{CZ}^{DBQA} = N_{CZ}^{VQE}(e) + n_{fval} \cdot n_{CZ}^{DBQA}. \tag{D.40}$$

D.6. Compiling of XXZ evolution

We now discuss the explicit circuit compilation for the diagonalized DBQA. For the XXZ model, the Hamiltonian is written as a sum of local terms:

$$\hat{H}_0 = \sum_{a=1}^{L} \hat{H}^{(a)} \tag{D.41}$$

where each term $\hat{H}^{(a)}$ acts on two qubits:

$$\hat{H}^{(a)} = \hat{X}_a \hat{X}_{a+1} + \hat{Y}_a \hat{Y}_{a+1} + \hat{Z}_a \hat{Z}_{a+1} \tag{D.42}$$

for $a < L$, and with periodic boundary conditions:

$$\hat{H}^{(L)} = \hat{X}_L \hat{X}_1 + \hat{Y}_L \hat{Y}_1 + \hat{Z}_L \hat{Z}_1. \tag{D.43}$$

Using M steps of the Trotter-Suzuki decomposition:

$$e^{-it\hat{H}_0} = \left(\prod_{a=1}^{L} e^{-i\frac{t}{M}\hat{H}^{(a)}} \right)^M + O(M^{-1}), \tag{D.44}$$

we achieve a circuit approximation with accuracy $O(t^2/M)$. Each two-qubit unitary $e^{-it/M\hat{H}^{(a)}}$ can be decomposed into a circuit of single-qubit rotations and 3 CNOT gates [189].

We now discuss generalizations. The Hamiltonian in Eq. (D.41) is arbitrary; only the final compilation step must be modified. The terms $\hat{H}^{(a)}$ can now act on more than two qubits. The resulting unitaries $e^{-it/M\hat{H}^{(a)}}$ are compiled using the quantum Shannon decomposition [190], which implements a K-qubit unitary with $O(4^K)$ CNOTs, a generically optimal scaling.

Alternatively, we can employ a higher-order Trotter-Suzuki decomposition. Consider a 2-qubit Hamiltonian on a line of even length L, and define $\hat{H}_0^{(\mathrm{o})} = \sum_{a=1}^{L/2} \hat{H}^{(2a-1)}$ and $\hat{H}_0^{(\mathrm{e})}$ such that all terms within each sum commute. Then,

$$e^{-it\hat{H}_0} = \left(e^{-it/(2M)h_0^{(\mathrm{o})}} e^{-it/M\hat{H}_0^{(\mathrm{e})}} e^{-it/(2M)\hat{H}_0^{(\mathrm{o})}} \right)^M + O(M^{-2}), \tag{D.45}$$

which achieves a better error scaling. Due to commutativity, we have the exact equality:

$$e^{-it/(2M)\hat{H}_0^{(\mathrm{o})}} = \prod_{a=1}^{L/2} e^{-it/M\hat{H}^{(2a+1)}}. \tag{D.46}$$

Each exponential in the product can be decomposed into CNOTs and single-qubit rotations using standard methods [189, 190].

D.6.1. Other 2-local models

The inclusion of a magnetic field is accommodated within this formalism by defining the local terms as:

$$\hat{H}^{(a)} = \hat{X}_a\hat{X}_{a+1} + \hat{Y}_a\hat{Y}_{a+1} + \hat{Z}_a\hat{Z}_{a+1} + B_aZ_a \tag{D.47}$$

for $a < L$. For periodic boundary conditions:

$$\hat{H}^{(L)} = \hat{X}_L\hat{X}_1 + \hat{Y}_L\hat{Y}_1 + \hat{Z}_L\hat{Z}_1 + B_L\hat{Z}_L, \tag{D.48}$$

while for open boundaries, $\hat{H}^{(L)} = 0$. The full Hamiltonian is then:

$$\hat{H}_0 = \hat{H}_{\mathrm{XXZ}} + \hat{H}(B) = \sum_{a=1}^{L} \hat{H}^{(a)}. \tag{D.49}$$

For the transverse-longitudinal field Ising model, we set:

$$\hat{H}^{(a)} = \hat{X}_a\hat{X}_{a+1} + B_a\hat{Z}_a + C_a\hat{X}_a, \tag{D.50}$$

with analogous boundary terms. Each two-qubit term $e^{-it/M\hat{H}^{(a)}}$ can be compiled into a circuit requiring 3 CNOT gates.

D.6.2. Special purpose compiling for the transverse-field Ising model

We now consider the special case of the transverse-field Ising model:

$$\hat{H}^{(a)} = \hat{X}_a\hat{X}_{a+1} + B_a\hat{Z}_a, \tag{D.51}$$

obtained by setting $C_a = 0$. Using the identities $\hat{X}_a\hat{X}_b = \mathrm{CNOT}(a,b)\hat{X}_a\mathrm{CNOT}(a,b)$ and $\hat{Z}_a = \mathrm{CNOT}(a,b)\hat{Z}_a\mathrm{CNOT}(a,b)$, we find:

$$\mathrm{CNOT}(a,b)\hat{H}^{(a)}\mathrm{CNOT}(a,b) = \hat{X}_a + B_a\hat{Z}_a. \tag{D.52}$$

This implies the evolution can be compiled as:

$$e^{-it\hat{H}^{(a)}} = \mathrm{CNOT}(a,b)e^{-it(\hat{X}_a+B_a\hat{Z}_a)}\mathrm{CNOT}(a,b), \tag{D.53}$$

requiring only 2 CNOT gates per term.

D.6.3. Compiling for the classical Ising model

For numerical efficiency, we use parametrizations with low compilation cost. Specifically, we define the classical Ising model:

$$\hat{H}(B,J) = \sum_{a=1}^{L}(B_aZ_a + J_{a,a+1}\hat{Z}_a\hat{Z}_{a+1}), \tag{D.54}$$

with $Z_{L+1} = Z_1$ for periodic boundaries. The evolution $e^{-it\hat{H}(B,J)}$ can be compiled using only 2 CNOT gates per interaction, which is more efficient than the general method [189]. Using the identity $Z_aZ_b = \mathrm{CNOT}(a,b)Z_b\mathrm{CNOT}(a,b)$, we get:

$$e^{-itZ_aZ_b} = \mathrm{CNOT}(a,b)e^{-itZ_b}\mathrm{CNOT}(a,b). \tag{D.55}$$

Since the model is commuting, $e^{-it\hat{H}(B,J=0)}$ consists of independent single-qubit rotations.

D.7. VQE×DBQA simulation results on Quantinuum

Experiments on IBM's superconducting chip showed that DBQA required error mitigation to outperform VQE; however, this may differ across platforms with distinct noise profiles, coherence times, and gate fidelities.

As shown in Table D.4, simulations on Quantinuum (*H1-Emulator*) indicate that one GCI step always improves over VQE, independent of noise and without post-processing. Adapting VQE and Hamiltonian simulation to native gates further benefits both algorithms.

While two GCI steps lower energy in the ideal case, realistic noise largely suppresses this gain, as DBQA circuit depth grows exponentially with GCI steps, compounding decoherence and gate errors.

D.8. VQE×DBQA results considering the J_1-J_2 model

In the main text, we reported results for the SU(2)-symmetric XXZ model with $\Delta = 0.5$ as penalty on the $\hat{Z}$ interactions. Here we extend the study to a more general Hamiltonian:

$$\hat{H}_{J_1\text{-}J_2} = J_1 \hat{H}_{\mathrm{XXZ}} + J_2 \sum_{i=1}^{L} (\hat{X}_i \hat{X}_{i+2} + \hat{Y}_i \hat{Y}_{i+2} + \hat{Z}_i \hat{Z}_{i+2}), \tag{D.56}$$

which includes next-nearest-neighbor couplings. We set $J_1 = 1$ and $J_2 = 0.2$, i.e., $J_2/J_1 = 0.2$, a regime close to the Berezinskii–Kosterlitz–Thouless transition at $J_2/J_1 = 0.24116$ [191, 192].

As in the XXZ case, we employ the Hamming-weight preserving ansatz of Fig. 11.3, since the same symmetries apply. The procedure follows that of the main text, but here DBQA is restricted to the single-commutator mode. Compilation of this model into a quantum circuit, as done in App. D.6, is left for future work. The corresponding results are summarized in Tab. D.5.

D.9. Details on relation to other methods

Many alternative VQE strategies exist, and we refer to reviews for detailed discussions of their performance [72, 73, 74]. In general, variational methods face training challenges (e.g., local minima or barren plateaus), so energy estimates often saturate before reaching the global minimum. In all such cases, the unitary $\hat{U}_{\theta^*}$ can be interfaced with DBQA via the input-generator rotation of Eq. (D.25), as in the main text.

We stress that VQE*x*DBQA differs conceptually from Ref. [132], which employs DBI methods as Riemannian flows to compute gradients of fixed parameterized circuits, i.e., a quantum version of Riemannian gradient descent. By contrast, our approach uses DBI methods to design new ansätze for parameterized quantum circuits.

Gradient-based methods are widely used to train VQE and can naturally be applied to DBQA. For instance, one can adopt a greedy 1-dimensional strategy, choosing DBI step durations s_k that minimize the cost in each DBR. More generally, gradient descent may be performed in the space of magnetic fields B_i or couplings $J_{i,j}$ to optimize the $\hat{D}_k$ operators from an initial guess. We have also explored non-gradient-based optimizers (see App. D.5).

Another direction is to replace the VQE unitaries $\hat{U}_\theta$ with Hartree–Fock warm-starts, where quantum compiling via Givens rotations has been experimentally demonstrated [85]. The key question is whether Hartree–Fock combined with DBQA yields improved ground-state approximations. More broadly, other quantum-chemistry ansätze can be interfaced with DBQA, provided explicit compiling is available.

Beyond direct energy minimization, some methods implement energy-lowering transformations. Quantum imaginary-time evolution (QITE) is an iterative algorithm where each step approximates the non-unitary update $|\psi_{k+1}\rangle = e^{-\tau\hat{H}_0}|\psi_k\rangle / |e^{-\tau\hat{H}_0}|\psi_k\rangle|$ with a unitary $\hat{U}_k$ such that $\hat{U}_k|\psi_k\rangle \approx |\psi_{k+1}\rangle$. The optimal form of $\hat{U}_k$ is unknown, necessitating variational methods for its construction [88, 89, 86].

While QITE avoids trainability issues by design, as each step is guaranteed to lower the energy, finding the correct unitary for longer durations becomes challenging due to limited cost-function resolution. In contrast, DBQA is a coherent, measurement-driven algorithm that does not rely on classical supervision for its parametrization. This suggests that a QITExDBQA interface (using QITE first) is likely preferable to DBQAxQITE. However, if QITE can be easily parameterized in the late stages of optimization, a QITExDBQAxQITE sequence could also be beneficial.

So far we have considered methods without auxiliary qubits. Allowing them brings in approaches based on quantum phase estimation (QPE), where interfacing with DBQA is also possible. Since QPE is generally more resource-intensive than DBQA, the natural ordering is DBQAxQPE rather than QPExDBQA, with hybrids such as VQExDBQAxQPE also conceivable.

QPE techniques can be systematically expressed within the qubitization framework, and QITE can likewise be formulated this way. In all such cases, the upper bound on the ground-state energy obtained via VQE*x*DBQA (or DBQAxQPE) can be complemented by a lower bound using Dual-VQE [193].

Finally, we comment on the relation between DBQA and the *quantum approximate optimization algorithm* (QAOA) [194, 72]. Eq. (D.10) highlights a resemblance to a 1-layer QAOA: in $\hat{V}_1$ a layer of single-qubit Z-rotations is followed by evolution under the input Hamiltonian and then by another rotation layer. The key difference is that in DBQA the

two rotation layers are inverses of each other, reflecting the guidance of the DBI equation via the reduced group commutator. This yields a constrained QAOA-like ansatz and ensures the monotonicity relation (D.10), providing analytical insight into diagonalization. Comparable relations for unconstrained QAOA are unknown, though Ref. [81] conjectured such connections might shed light on optimized QAOA performance.

Beyond these superficial similarities, the methods diverge. Ref. [194] applied a QAOA ansatz with native Hamiltonian evolutions to approximate the ground state of another model. In contrast, DBQA is not limited to ground states but can target arbitrary eigenstates. The distinction is even clearer beyond the first step: e.g., in $\hat{V}_2$ (Eq. (D.33)) DBQA incorporates both forward and backward evolutions under the Hamiltonian. While such nesting may be difficult to realize on analog simulators, it underpins DBQA's convergence guarantees, something not known for QAOA.

D.9.1. Overview of currently available circuit depth

Following the overview above, most methods for preparing ground states on existing quantum hardware suffer one of two severe problems. Variational methods are limited by the resolution of the cost function which is at the root of their operation. This limits the circuit depth that is meaningful in practice to about a dozen entangling layers. QPE-based methods are the opposite in that to become meaningful they need a large circuit depth.

As a specific example, we consider Ref. [195] which compared the performance of QPE for physical and logical qubits. The algorithmic performance in Ref. [195] was obtained by using $N \approx 920$ CZ gates for $L = 8$ qubits and the 2-qubit gate fidelity was $p_e \approx 2 \times 10^{-3}$.

We can use these experimental results to get ballpark figures for currently available depths. Upcoming devices are set to reach $p'_e \approx 5 \times 10^{-4}$ [196] so assuming each CZ gate fails independently of the others we can estimate the number N' of gates that can be used meaningfully by solving heuristically

$$(1-p_e)^N = (1-p'_e)^{N'} \tag{D.57}$$

This equation gives 16% success probability which in the three-nines fidelity regime p'_e would appear for $N' \approx 3690$ CZ gates. From these estimates, we see that there exists quantum hardware which operates meaningfully for circuit depths of about 100 CZ gates per qubit and this is set to even larger values in the very close future. This means that 1 and 2 GCI steps are already feasible for implementation on existing noisy intermediate-scale quantum devices.

Qubits	VQE training		1 GCI step		2 GCI steps		3 GCI steps	
	Depth	$\mathcal{F}$	Depth	$\mathcal{F}$	Depth	$\mathcal{F}$	Depth	$\mathcal{F}$
4	16	1	95	1	490	1	2465	1
6	15	1	88	1	457	1	2298	1
8	16	0.947 ± 0.024	95	0.9977 ± 0.0017	490	0.9990 ± 0.0008	2465	0.9995 ± 0.0005
10	16	0.889 ± 0.016	95	0.9928 ± 0.0012	490	0.9972 ± 0.0012	2465	0.9987 ± 0.0005
12	15	0.600 ± 0.091	92	0.919 ± 0.067	473	0.941 ± 0.048	2382	0.956 ± 0.037

Table D.1.: Fidelity lower bound $\mathcal{F}$ (Eq. (D.36)) for the XXZ model (Eq. (11.7)) with $L \in 4, 6, 8, 10, 12$ qubits and $\Delta = 0.5$. Results are shown for pure VQE approximations and for VQExDBQA results obtained from compiled GCI circuits. The left sub-column for each configuration displays the corresponding circuit depth. The reported values and their uncertainties represent the median and median absolute deviation, respectively, computed from a sample of ten independent runs with different initial conditions.

Layers	Warm-start	1 GCI step		2 GCI steps		Long VQE training	
	$1-\tilde{E}_0/E_0$	$1-\tilde{E}_0/E_0$		$1-\tilde{E}_0/E_0$		$1-\tilde{E}_0/E_0$	
7	0.023 ± 0.002	0.013 ± 0.0007		0.011 ± 0.0003		0.006 ± 0.002	
8	0.005 ± 0.003	0.0003 ± 0.0002		0.00013 ± 0.00007		0.004 ± 0.002	
9	0.005 ± 0.003	0.0003 ± 0.0002		0.00018 ± 0.00006		0.0030 ± 0.0003	
	Depth	Cumulative cost	Depth	Cumulative cost	Depth	Cumulative cost	Depth	Cumulative cost
7	7	12.18×10^7	50	12.25×10^7	265	12.52×10^7	7	20.03×10^7
8	8	15.84×10^7	55	15.91×10^7	290	16.21×10^7	8	26.4×10^7
9	9	$19,98 \times 10^7$	60	20.06×10^7	315	20.39×10^7	9	33.3×10^7

Table D.2.: *(above)* Energy approximation ratio for the XXZ model of Eq. (11.7) with $L = 10$ qubits, and $\Delta = 0.5$. The estimates with their uncertainties were calculated using the median and the median absolute deviation of a sample of results obtained by repeating the execution fifty times with different initial conditions. *(below)* Circuit depth expressed as number of CZ gates per qubit, alongside with cumulative number of CZ gates used to reach $\tilde{E}_0$ (See App. D.5). Warm-start VQE approximations (3000 epochs of training) are presented alongside VQExDBQA results, executed considering compiled GCI circuits. Longer VQE training (5000 epochs) is reported in the last column of the table.

Layers	Warm-start	1 GCI step	2 GCI steps	Long VQE training
7	0.67 ± 0.03	0.82 ± 0.05	0.85 ± 0.05	0.92 ± 0.03
8	0.93 ± 0.04	0.995 ± 0.002	0.998 ± 0.001	0.94 ± 0.03
9	0.93 ± 0.02	0.996 ± 0.001	0.998 ± 0.001	0.96 ± 0.004

Table D.3.: Fidelity lower bound $\mathcal{F}$ (see Eq. (D.36)) for the XXZ model of Eq. (11.7) with $L = 10$ qubits, and $\Delta = 0.5$ extending results of Tab. D.2.

	Warm-start	1 GCI step	2 CGI steps
statevector	-14.16	-14.63	-14.72
H1-LE	-14.19 ± 0.16	-14.63 ± 0.10	-14.70 ± 0.11
H1-Emulator	-14.14 ± 0.15	-14.59 ± 0.23	-14.07 ± 0.10

Table D.4.: Simulation results on Quantinuum's H1 emulators showing the energy expectation of the Heisenberg XXZ model with $L = 10$ and $\Delta = 0.5$ after a 2-layer VQE and GCI steps. Both the VQE and hamiltonian simulation are adapted to the platform's supported native gates.

Layers	Warm-start	1 DBI step	2 DBI steps	3 DBI steps	Long VQE training
3	0.033 ± 0.005	0.020 ± 0.005	0.016 ± 0.005	0.014 ± 0.004	0.026 ± 0.006
4	0.017 ± 0.007	0.006 ± 0.005	0.003 ± 0.003	0.002 ± 0.002	0.010 ± 0.004
5	0.011 ± 0.007	0.002 ± 0.002	0.0008 ± 0.0008	0.0004 ± 0.0004	0.005 ± 0.003
6	0.009 ± 0.006	0.002 ± 0.001	0.0007 ± 0.0007	0.0004 ± 0.0004	0.005 ± 0.003

Table D.5.: Relative difference between approximated energy $\tilde{E}_0$ and the target ground state value E_0 for the J_1-J_2 model of Eq. (D.56) with $L = 10$ qubits, $J_1 = 1$ and $J_2 = 0.2$ together with the cumulative number of CZ gates, N_{CZ}, used to reach $\tilde{E}_0$. The estimates with their uncertainties were calculated using the median and the median absolute deviation of a sample of results obtained by repeating the execution fifty times with different initial conditions. Warm-start VQE approximations (500 epochs of training) are presented alongside VQExDBQA results, executed considering BDI unitaries. Longer VQE trainings (2000 epochs) are reported in the last column of the table.

List of Figures

List of Tables

Bibliography

[1] Riccardo Manenti and Mario Motta. *Quantum Information Science*. Oxford University Press, Aug. 2023. ISBN: 978-0-19-878748-8.

[2] John M. Martinis and Kevin Osborne. *Superconducting Qubits and the Physics of Josephson Junctions*. 2004. arXiv: cond-mat/0402415 [cond-mat.supr-con]. URL: https://arxiv.org/abs/cond-mat/0402415.

[3] S.E. Rasmussen et al. "Superconducting Circuit Companion—an Introduction with Worked Examples". In: *PRX Quantum* 2 (4 Dec. 2021), p. 040204. DOI: 10.1103/PRXQuantum.2.040204. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.040204.

[4] Jens Koch et al. "Charge-insensitive qubit design derived from the Cooper pair box". In: *Phys. Rev. A* 76 (4 Oct. 2007), p. 042319. DOI: 10.1103/PhysRevA.76.042319. URL: https://link.aps.org/doi/10.1103/PhysRevA.76.042319.

[5] P. Krantz et al. "A quantum engineer's guide to superconducting qubits". In: *Applied Physics Reviews* 6.2 (June 2019). ISSN: 1931-9401. DOI: 10.1063/1.5089550. URL: http://dx.doi.org/10.1063/1.5089550.

[6] Jens Koch et al. "Charge-insensitive qubit design derived from the Cooper pair box". In: *Physical Review A* 76.4 (Oct. 2007). DOI: 10.1103/physreva.76.042319. URL: https://doi.org/10.1103%2Fphysreva.76.042319.

[7] Alexandre Blais et al. "Circuit quantum electrodynamics". In: *Reviews of Modern Physics* 93.2 (May 2021). ISSN: 1539-0756. DOI: 10.1103/revmodphys.93.025005. URL: http://dx.doi.org/10.1103/RevModPhys.93.025005.

[8] Stavros Efthymiou et al. "Qibo: a framework for quantum simulation with hardware acceleration". In: *Quantum Science and Technology* 7.1 (Dec. 2021), p. 015018. DOI: 10.1088/2058-9565/ac39f5.

[9] Stavros Efthymiou et al. *qiboteam/qibolab: Qibolab 0.0.2*. Version v0.0.2. Mar. 2023. DOI: 10.5281/zenodo.7748527. URL: https://doi.org/10.5281/zenodo.7748527.

[10] Andrea Pasquale et al. *Qibocal: an open-source framework for calibration of self-hosted quantum devices*. 2024. arXiv: 2410.00101 [quant-ph]. URL: https://arxiv.org/abs/2410.00101.

[11] Rodolfo Carobene et al. “Qibosoq: an open-source framework for quantum circuit RFSoC programming”. In: *Quantum Science and Technology* 10.3 (Apr. 2025), p. 035010. DOI: 10.1088/2058-9565/adcd97. URL: https://doi.org/10.1088/2058-9565/adcd97.

[12] Leandro Stefanazzi et al. “The QICK (Quantum Instrumentation Control Kit): Readout and control for qubits and detectors”. In: *Review of Scientific Instruments* 93.4 (Apr. 2022). ISSN: 1089-7623. DOI: 10.1063/5.0076249. URL: http://dx.doi.org/10.1063/5.0076249.

[13] John Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (Aug. 2018), p. 79. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331%2Fq-2018-08-06-79.

[14] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[15] Travis E. Oliphant. *Guide to NumPy*. Trelgol, 2006.

[16] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[17] S. Carrazza et al. “An open-source modular framework for quantum computing”. In: *Journal of Physics: Conference Series* 2438.1 (Feb. 2023), p. 012148. DOI: 10.1088/1742-6596/2438/1/012148. URL: https://doi.org/10.1088%2F1742-6596%2F2438%2F1%2F012148.

[18] Carlos Bravo-Prieto et al. “Style-based quantum generative adversarial networks for Monte Carlo events”. In: *Quantum* 6 (Aug. 2022), p. 777. DOI: 10.22331/q-2022-08-17-777. URL: https://doi.org/10.22331%2Fq-2022-08-17-777.

[19] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (1986), pp. 533–536. URL: https://doi.org/10.1038/323533a0.

[20] Stavros Efthymiou et al. “Quantum simulation with just-in-time compilation”. In: *Quantum* 6 (Sept. 2022), p. 814. DOI: 10.22331/q-2022-09-22-814. URL: https://doi.org/10.22331%2Fq-2022-09-22-814.

[21] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. “Numba: A llvm-based python jit compiler”. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 2015, pp. 1–6. DOI: 10.1145/2833157.2833162. URL: http://dx.doi.org/10.1145/2833157.2833162.

[22] Ryosuke Okuta et al. “CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations”. In: *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*. 2017. URL: http://learningsys.org/nips17/assets/papers/paper_16.pdf.

[23] The cuQuantum development team. *cuQuantum*. Version v23.03.0. If you use this software, please cite it as below. Apr. 2023. DOI: 10.5281/zenodo.7806810. URL: https://doi.org/10.5281/zenodo.7806810.

[24] Jacob Biamonte and Ville Bergholm. *Tensor Networks in a Nutshell*. 2017. DOI: 10.48550/arXiv.1708.00006. URL: https://doi.org/10.48550/arXiv.1708.00006.

[25] Xiao Yuan et al. “Quantum Simulation with Hybrid Tensor Networks”. In: *Physical Review Letters* 127.4 (July 2021). DOI: 10.1103/physrevlett.127.040501. URL: https://doi.org/10.1103%2Fphysrevlett.127.040501.

[26] William Huggins et al. “Towards quantum machine learning with tensor networks”. In: *Quantum Science and Technology* 4.2 (Jan. 2019), p. 024001. DOI: 10.1088/2058-9565/aaea94. URL: https://dx.doi.org/10.1088/2058-9565/aaea94.

[27] Román Orús. “A practical introduction to tensor networks: Matrix product states and projected entangled pair states”. In: *Annals of Physics* 349 (2014), pp. 117–158. ISSN: 0003-4916. DOI: 10.1016/j.aop.2014.06.013. URL: https://doi.org/10.1016/j.aop.2014.06.013.

[28] Jacob Biamonte. *Lectures on Quantum Tensor Networks*. 2020. DOI: 10.48550/arXiv.1912.10049. URL: https://doi.org/10.48550/arXiv.1912.10049.

[29] Yvonne Y. Gao et al. “Practical Guide for Building Superconducting Quantum Devices”. In: *PRX Quantum* 2 (4 Nov. 2021), p. 040202. DOI: 10.1103/PRXQuantum.2.040202. URL: https://doi.org/10.1103/PRXQuantum.2.040202.

[30] D. Leibfried et al. “Quantum dynamics of single trapped ions”. In: *Rev. Mod. Phys.* 75 (1 Mar. 2003), pp. 281–324. DOI: 10.1103/RevModPhys.75.281. URL: https://doi.org/10.1103/RevModPhys.75.281.

[31] Loïc Henriet et al. “Quantum computing with neutral atoms”. In: *Quantum* 4 (Sept. 2020), p. 327. DOI: 10.22331/q-2020-09-21-327. URL: https://doi.org/10.22331%2Fq-2020-09-21-327.

[32] Thomas Alexander et al. “Qiskit pulse: programming quantum computers through the cloud with pulses”. In: *Quantum Science and Technology* 5.4 (Aug. 2020), p. 044006. ISSN: 2058-9565. DOI: 10.1088/2058-9565/aba404. URL: http://dx.doi.org/10.1088/2058-9565/aba404.

[33] Henrique Silvério et al. “Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays”. In: *Quantum* 6 (Jan. 2022), p. 629. ISSN: 2521-327X. DOI: 10.22331/q-2022-01-24-629. URL: http://dx.doi.org/10.22331/q-2022-01-24-629.

[34] ZurichInstruments. https://www.zhinst.com/others/en/quantum-computing-systems/labone-q. 2023.

[35] Lior Ella et al. *Quantum-classical processing and benchmarking at the pulse-level*. 2023. DOI: 10.48550/arXiv.2303.03816. URL: https://doi.org/10.48550/arXiv.2303.03816.

[36] Qblox. https://qblox-qblox-instruments.readthedocs-hosted.com/en/master/. 2023.

[37] Jens Hedegaard Nielsen et al. *QCoDeS/Qcodes: QCoDeS 0.43.0*. Version v0.43.0. Jan. 2024. DOI: 10.5281/zenodo.10459033. URL: https://doi.org/10.5281/zenodo.10459033.

[38] F. Motzoi et al. “Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits”. In: *Phys. Rev. Lett.* 103 (11 Sept. 2009), p. 110501. DOI: 10.1103/PhysRevLett.103.110501. URL: https://doi.org/10.1103/PhysRevLett.103.110501.

[39] Johannes Heinsoo et al. “Rapid High-fidelity Multiplexed Readout of Superconducting Qubits”. In: *Phys. Rev. Appl.* 10 (3 Sept. 2018), p. 034040. DOI: 10.1103/PhysRevApplied.10.034040. URL: https://doi.org/10.1103/PhysRevApplied.10.034040.

[40] Yilun Xu et al. “Automatic Qubit Characterization and Gate Optimization with QubiC”. In: *ACM Transactions on Quantum Computing* 4.1 (Oct. 2022). ISSN: 2643-6809. DOI: 10.1145/3529397. URL: https://doi.org/10.1145/3529397.

[41] Julian Kelly et al. *Physical qubit calibration on a directed acyclic graph*. 2018. DOI: 10.48550/arXiv.1803.03226. URL: https://doi.org/10.48550/arXiv.1803.03226.

[42] Qibolab: Platform creation. https://qibo.science/qibolab/stable/tutorials/lab.html.

[43] Qibolab: Platform serialization. https://qibo.science/qibolab/stable/api-reference/qibolab.html#module-qibolab.serialize.

[44] Easwar Magesan and Jay M. Gambetta. “Effective Hamiltonian models of the cross-resonance gate”. In: *Phys. Rev. A* 101 (5 May 2020), p. 052308. DOI: 10.1103/PhysRevA.101.052308. URL: https://link.aps.org/doi/10.1103/PhysRevA.101.052308.

[45] J.R. Johansson, P.D. Nation, and Franco Nori. “QuTiP 2: A Python framework for the dynamics of open quantum systems”. In: *Computer Physics Communications* 184.4 (2013), pp. 1234–1240. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2012.11.019. URL: https://www.sciencedirect.com/science/article/pii/S0010465512003955.

[46] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/google/jax.

[47] Julian Kelly et al. *Physical qubit calibration on a directed acyclic graph*. 2018. DOI: 10.48550/ARXIV.1803.03226. arXiv: 1803.03226 [quant-ph].

[48] David C. McKay et al. "Efficient gates for quantum computing". In: *Physical Review A* 96.2 (Aug. 2017). ISSN: 2469-9934. DOI: 10.1103/physreva.96.022330. URL: http://dx.doi.org/10.1103/PhysRevA.96.022330.

[49] S. Probst et al. "Efficient and robust analysis of complex scattering data under noise in microwave resonators". In: *Review of Scientific Instruments* 86.2 (Feb. 2015), p. 024706. ISSN: 0034-6748. DOI: 10.1063/1.4907935. eprint: https://pubs.aip.org/aip/rsi/article-pdf/doi/10.1063/1.4907935/15732678/024706_1_online.pdf. URL: https://doi.org/10.1063/1.4907935.

[50] D. I. Schuster et al. "ac Stark Shift and Dephasing of a Superconducting Qubit Strongly Coupled to a Cavity Field". In: *Physical Review Letters* 94.12 (Mar. 2005). ISSN: 1079-7114. DOI: 10.1103/physrevlett.94.123602. URL: http://dx.doi.org/10.1103/PhysRevLett.94.123602.

[51] Yvonne Y. Gao et al. "Practical Guide for Building Superconducting Quantum Devices". In: *PRX Quantum* 2 (4 Nov. 2021), p. 040202. DOI: 10.1103/PRXQuantum.2.040202. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.040202.

[52] Sarah Sheldon et al. "Characterizing errors on qubit operations via iterative randomized benchmarking". In: *Phys. Rev. A* 93 (1 Jan. 2016), p. 012301. DOI: 10.1103/PhysRevA.93.012301. URL: https://link.aps.org/doi/10.1103/PhysRevA.93.012301.

[53] Colm A. Ryan et al. "Tomography via correlation of noisy measurement records". In: *Physical Review A* 91.2 (2015). ISSN: 1094-1622. DOI: 10.1103/physreva.91.022118. URL: http://dx.doi.org/10.1103/PhysRevA.91.022118.

[54] Evan Jeffrey et al. "Fast Accurate State Measurement with Superconducting Qubits". In: *Physical Review Letters* 112.19 (May 2014). ISSN: 1079-7114. DOI: 10.1103/physrevlett.112.190504. URL: http://dx.doi.org/10.1103/PhysRevLett.112.190504.

[55] C. C. Bultink et al. "General method for extracting the quantum efficiency of dispersive qubit readout in circuit QED". In: *Applied Physics Letters* 112.9 (Feb. 2018). ISSN: 1077-3118. DOI: 10.1063/1.5015954. URL: http://dx.doi.org/10.1063/1.5015954.

[56] D. T. McClure et al. “Rapid Driven Reset of a Qubit Readout Resonator”. In: *Physical Review Applied* 5.1 (Jan. 2016). ISSN: 2331-7019. DOI: 10.1103/physrevapplied.5.011001. URL: http://dx.doi.org/10.1103/PhysRevApplied.5.011001.

[57] Akel Hashim et al. “Practical Introduction to Benchmarking and Characterization of Quantum Computers”. In: *PRX Quantum* 6.3 (Aug. 2025). ISSN: 2691-3399. DOI: 10.1103/prxquantum.6.030202. URL: http://dx.doi.org/10.1103/PRXQuantum.6.030202.

[58] Joseph Emerson, Robert Alicki, and Karol Życzkowski. “Scalable noise estimation with random unitary operators”. In: *Journal of Optics B: Quantum and Semiclassical Optics* 7.10 (Sept. 2005), S347. DOI: 10.1088/1464-4266/7/10/021. URL: https://dx.doi.org/10.1088/1464-4266/7/10/021.

[59] L. DiCarlo et al. “Demonstration of two-qubit algorithms with a superconducting quantum processor”. In: *Nature* 460.7252 (June 2009), pp. 240–244. ISSN: 1476-4687. DOI: 10.1038/nature08121. URL: http://dx.doi.org/10.1038/nature08121.

[60] John M. Martinis and Michael R. Geller. “Fast adiabatic qubit gates using only σ_z control”. In: *Phys. Rev. A* 90 (2 Aug. 2014), p. 022307. DOI: 10.1103/PhysRevA.90.022307. URL: https://link.aps.org/doi/10.1103/PhysRevA.90.022307.

[61] L. DiCarlo et al. “Preparation and measurement of three-qubit entanglement in a superconducting circuit”. In: *Nature* 467.7315 (Sept. 2010), pp. 574–578. ISSN: 1476-4687. DOI: 10.1038/nature09416. URL: http://dx.doi.org/10.1038/nature09416.

[62] M. Rol et al. “Time-domain characterization and correction of on-chip distortion of control pulses in a quantum processor”. In: *Applied Physics Letters* 116 (Feb. 2020), p. 054001. DOI: 10.1063/1.5133894.

[63] Christoph Hellings et al. *Calibrating Magnetic Flux Control in Superconducting Circuits by Compensating Distortions on Time Scales from Nanoseconds up to Tens of Microseconds*. 2025. arXiv: 2503.04610 [quant-ph]. URL: https://arxiv.org/abs/2503.04610.

[64] V. Negirneac et al. “High-Fidelity Controlled-*Z* Gate with Maximal Intermediate Leakage Operating at the Speed Limit in a Superconducting Quantum Processor”. In: *Phys. Rev. Lett.* 126 (22 June 2021), p. 220502. DOI: 10.1103/PhysRevLett.126.220502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.126.220502.

[65] M. A. Rol et al. “Fast, High-Fidelity Conditional-Phase Gate Exploiting Leakage Interference in Weakly Anharmonic Superconducting Qubits”. In: *Phys. Rev. Lett.* 123 (12 Sept. 2019), p. 120502. DOI: 10.1103/PhysRevLett.123.120502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.123.120502.

[66] Yulong Dong, Lin Lin, and Yu Tong. “Ground-State Preparation and Energy Estimation on Early Fault-Tolerant Quantum Computers via Quantum Eigenvalue Transformation of Unitary Matrices”. In: *PRX Quantum* 3.4 (Oct. 2022). ISSN: 2691-3399. DOI: 10.1103/prxquantum.3.040305. URL: http://dx.doi.org/10.1103/PRXQuantum.3.040305.

[67] Lin Lin and Yu Tong. “Heisenberg-Limited Ground-State Energy Estimation for Early Fault-Tolerant Quantum Computers”. In: *PRX Quantum* 3 (1 Feb. 2022), p. 010318. DOI: 10.1103/PRXQuantum.3.010318. URL: https://link.aps.org/doi/10.1103/PRXQuantum.3.010318.

[68] Yimin Ge, Jordi Tura, and J Ignacio Cirac. “Faster ground state preparation and high-precision ground energy estimation with fewer qubits”. In: *Journal of Mathematical Physics* 60.2 (2019), p. 022202. DOI: https://doi.org/10.1063/1.5027484.

[69] Oriel Kiss et al. “Early Fault-Tolerant Quantum Algorithms in Practice: Application to Ground-State Energy Estimation”. In: *Quantum* 9 (Apr. 2025), p. 1682. ISSN: 2521-327X. DOI: 10.22331/q-2025-04-01-1682. URL: https://doi.org/10.22331/q-2025-04-01-1682.

[70] Harper R Grimsley et al. “An adaptive variational algorithm for exact molecular simulations on a quantum computer”. In: *Nature Communications* 10.1 (2019), pp. 1–9. DOI: 10.1038/s41467-019-10988-2. URL: https://www.nature.com/articles/s41467-019-10988-2.

[71] Alberto Peruzzo et al. “A variational eigenvalue solver on a photonic quantum processor”. In: *Nature Communications* 5.1 (July 2014). ISSN: 2041-1723. DOI: 10.1038/ncomms5213. URL: http://dx.doi.org/10.1038/ncomms5213.

[72] Kishor Bharti et al. “Noisy intermediate-scale quantum algorithms”. In: *Rev. Mod. Phys.* 94 (1 Feb. 2022), p. 015004. DOI: 10.1103/RevModPhys.94.015004. URL: https://link.aps.org/doi/10.1103/RevModPhys.94.015004.

[73] Marco Cerezo et al. “Variational quantum algorithms”. In: *Nature Reviews Physics* 3.9 (2021), pp. 625–644.

[74] Jules Tilly et al. “The variational quantum eigensolver: a review of methods and best practices”. In: *Physics Reports* 986 (2022), pp. 1–128.

[75] Eric R Anschuetz and Bobak T Kiani. “Beyond Barren Plateaus: Quantum Variational Algorithms Are Swamped With Traps”. In: *Nature Communications* 13.1 (2022), p. 7760. DOI: 10.1038/s41467-022-35364-5. URL: https://doi.org/10.1038/s41467-022-35364-5.

[76] Jarrod R. McClean et al. “Barren plateaus in quantum neural network training landscapes”. In: *Nature Communications* 9.1 (Nov. 2018). ISSN: 2041-1723. DOI: 10.1038/s41467-018-07090-4. URL: http://dx.doi.org/10.1038/s41467-018-07090-4.

[77] Lennart Bittel and Martin Kliesch. “Training Variational Quantum Algorithms Is NP-Hard”. In: *Phys. Rev. Lett.* 127 (12 Sept. 2021), p. 120502. DOI: 10.1103/PhysRevLett.127.120502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.127.120502.

[78] Daniel Stilck Franca and Raul Garcia-Patron. “Limitations of optimization algorithms on noisy quantum devices”. In: *Nature Physics* 17.11 (2021), pp. 1221–1227. DOI: https://doi.org/10.1038/s41567-021-01356-3.

[79] Martin Larocca et al. *A Review of Barren Plateaus in Variational Quantum Computing*. 2024. arXiv: 2405.00781 [quant-ph]. URL: https://arxiv.org/abs/2405.00781.

[80] M Cerezo et al. “Does provable absence of barren plateaus imply classical simulability? Or, why we need to rethink variational quantum computing”. In: *arXiv preprint arXiv:2312.09121* (2023). URL: https://arxiv.org/abs/2312.09121.

[81] Marek Gluza. “Double-bracket quantum algorithms for diagonalization”. In: *Quantum* 8 (Apr. 2024), p. 1316. ISSN: 2521-327X. DOI: 10.22331/q-2024-04-09-1316. URL: http://dx.doi.org/10.22331/q-2024-04-09-1316.

[82] Uwe Helmke and John B. Moore. *Optimization and Dynamical Systems*. Springer London, 1994. DOI: https://doi.org/10.1007/978-1-4471-3467-1.

[83] John B. Moore, Robert E. Mahony, and Uwe Helmke. “Numerical Gradient Algorithms for Eigenvalue and Singular Value Calculations”. In: *SIAM J. Matrix Anal. Appl.* 15 (1994), pp. 881–902. URL: https://api.semanticscholar.org/CorpusID:17253271.

[84] Steven Thomas Smith. *Geometric optimization methods for adaptive filtering*. Harvard University, 1993. DOI: https://doi.org/10.48550/arXiv.1305.1886.

[85] Frank Arute et al. “Hartree-Fock on a superconducting qubit quantum computer”. In: *Science* 369.6507 (2020), pp. 1084–1089. DOI: 10.1126/science.abb9811. URL: https://science.sciencemag.org/content/369/6507/1084.

[86] Mario Motta et al. “Determining eigenstates and thermal states on a quantum computer using quantum imaginary time evolution”. In: *Nature Physics* 16.2 (2020), pp. 205–210. DOI: 10.1038/s41567-019-0704-4. URL: https://www.nature.com/articles/s41567-019-0704-4.

[87] Hirofumi Nishi, Taichi Kosugi, and Yu-ichiro Matsushita. “Implementation of quantum imaginary-time evolution method on NISQ devices by introducing nonlocal approximation”. In: *npj Quantum Information* 7.1 (June 2021), p. 85. ISSN: 2056-6387. DOI: 10.1038/s41534-021-00409-y. URL: https://doi.org/10.1038/s41534-021-00409-y.

[88] Sam McArdle et al. “Variational ansatz-based quantum simulation of imaginary time evolution”. In: *npj Quantum Information* 5.1 (Sept. 2019), p. 75. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0187-2. URL: https://doi.org/10.1038/s41534-019-0187-2.

[89] Niladri Gomes et al. “Adaptive Variational Quantum Imaginary Time Evolution Approach for Ground State Preparation”. In: *Advanced Quantum Technologies* 4.12 (2021), p. 2100114. DOI: https://doi.org/10.1002/qute.202100114.

[90] Franz Wegner. “Flow-equations for Hamiltonians”. In: *Annalen der Physik* 506.2 (1994), pp. 77–91. DOI: https://doi.org/10.1002/andp.19945060203.

[91] Stanisław D. Głazek and Kenneth G. Wilson. “Renormalization of Hamiltonians”. In: *Phys. Rev. D* 48 (12 Dec. 1993), pp. 5863–5872. DOI: 10.1103/PhysRevD.48.5863. URL: https://link.aps.org/doi/10.1103/PhysRevD.48.5863.

[92] Franz Wegner. “Flow equations and normal ordering: a survey”. In: *Journal of Physics A: Mathematical and General* 39.25 (2006), p. 8221. DOI: 10.1088/0305-4470/39/25/s29.

[93] David Pekker et al. “Fixed Points of Wegner-Wilson Flows and Many-Body Localization”. In: *Phys. Rev. Lett.* 119 (7 Aug. 2017), p. 075701. DOI: 10.1103/PhysRevLett.119.075701. URL: https://link.aps.org/doi/10.1103/PhysRevLett.119.075701.

[94] Zoë Holmes et al. “Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus”. In: *PRX Quantum* 3.1 (Jan. 2022). ISSN: 2691-3399. DOI: 10.1103/prxquantum.3.010313. URL: http://dx.doi.org/10.1103/PRXQuantum.3.010313.

[95] Yu-An Chen et al. “Efficient product formulas for commutators and applications to quantum simulation”. In: *Phys. Rev. Res.* 4 (1 Mar. 2022), p. 013191. DOI: 10.1103/PhysRevResearch.4.013191. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.4.013191.

[96] Christopher M Dawson and Michael A Nielsen. “The Solovay-Kitaev algorithm”. In: *Quantum Information & Computation* 6.1 (2006), pp. 81–95. DOI: https://doi.org/10.48550/arXiv.quant-ph/0505030.

[97] Andrew M. Childs and Yuan Su. “Nearly Optimal Lattice Simulation by Product Formulas”. In: *Phys. Rev. Lett.* 123 (5 Aug. 2019), p. 050503. DOI: 10.1103/PhysRevLett.123.050503. URL: https://link.aps.org/doi/10.1103/PhysRevLett.123.050503.

[98] Earl Campbell. “Random Compiler for Fast Hamiltonian Simulation”. In: *Phys. Rev. Lett.* 123 (7 Aug. 2019), p. 070503. DOI: 10.1103/PhysRevLett.123.070503. URL: https://link.aps.org/doi/10.1103/PhysRevLett.123.070503.

[99] Pierre Nataf and Frederic Mila. “Exact Diagonalization of Heisenberg SU(*N*) Models”. In: *Phys. Rev. Lett.* 113 (12 Sept. 2014), p. 127204. DOI: 10.1103/PhysRevLett.113.127204. URL: https://link.aps.org/doi/10.1103/PhysRevLett.113.127204.

[100] M. Raissi, P. Perdikaris, and G.E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[101] Gian-Luca R Anselmetti et al. “Local, expressive, quantum-number-preserving VQE ansätze for fermionic systems”. In: *New Journal of Physics* 23.11 (Nov. 2021), p. 113010. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ac2cb3. URL: http://dx.doi.org/10.1088/1367-2630/ac2cb3.

[102] Juan Miguel Arrazola et al. “Universal quantum circuits for quantum chemistry”. In: *Quantum* 6 (June 2022), p. 742. ISSN: 2521-327X. DOI: 10.22331/q-2022-06-20-742. URL: http://dx.doi.org/10.22331/q-2022-06-20-742.

[103] Sonika Johri et al. “Nearest centroid classification on a trapped ion quantum computer”. In: *npj Quantum Information* 7 (1 Aug. 2021), p. 122. DOI: 10.1038/s41534-021-00456-5. URL: https://doi.org/10.1038/s41534-021-00456-5.

[104] Iordanis Kerenidis and Anupam Prakash. *Quantum machine learning with subspace states*. 2022. arXiv: 2202.00054 [quant-ph]. URL: https://arxiv.org/abs/2202.00054.

[105] Nishant Jain et al. “Quantum Fourier networks for solving parametric PDEs”. In: *Quantum Science and Technology* 9.3 (May 2024), p. 035026. DOI: 10.1088/2058-9565/ad42ce. URL: https://dx.doi.org/10.1088/2058-9565/ad42ce.

[106] Michael Ragone et al. *Representation Theory for Geometric Quantum Machine Learning*. 2023. arXiv: 2210.07980 [quant-ph]. URL: https://arxiv.org/abs/2210.07980.

[107] Leo Monbroussou et al. “Trainability and Expressivity of Hamming-Weight Preserving Quantum Circuits for Machine Learning”. In: *arXiv preprint arXiv:2309.15547* (2023). URL: https://arxiv.org/abs/2309.15547.

[108] Cenk Tüysüz et al. *Symmetry breaking in geometric quantum machine learning in the presence of noise*. 2024. arXiv: 2401.10293 [quant-ph]. URL: https://arxiv.org/abs/2401.10293.

[109] David Raveh and Rafael I. Nepomechie. *Estimating Bethe roots with VQE*. 2024. arXiv: 2404.18244 [quant-ph]. URL: https://arxiv.org/abs/2404.18244.

[110] Renato M. S. Farias et al. *Quantum encoder for fixed Hamming-weight subspaces*. 2024. arXiv: 2405.20408 [quant-ph]. URL: https://arxiv.org/abs/2405.20408.

[111] El Amine Cherrat et al. “Quantum Vision Transformers”. In: *Quantum* 8 (Feb. 2024), p. 1265. ISSN: 2521-327X. DOI: 10.22331/q-2024-02-22-1265. URL: http://dx.doi.org/10.22331/q-2024-02-22-1265.

[112] Giulio Crognaletti et al. *Equivariant Variational Quantum Eigensolver to detect Phase Transitions through Energy Level Crossings*. 2024. arXiv: 2403.07100 [quant-ph]. URL: https://arxiv.org/abs/2403.07100.

[113] Oriel Kiss et al. “Quantum computing of the ^{6}Li nucleus via ordered unitary coupled clusters”. In: *Phys. Rev. C* 106 (3 Sept. 2022), p. 034325. DOI: 10.1103/PhysRevC.106.034325. URL: https://link.aps.org/doi/10.1103/PhysRevC.106.034325.

[114] Michele Grossi et al. “Finite-size criticality in fully connected spin models on superconducting quantum hardware”. In: *Phys. Rev. E* 107 (2 Feb. 2023), p. 024113. DOI: 10.1103/PhysRevE.107.024113. URL: https://link.aps.org/doi/10.1103/PhysRevE.107.024113.

[115] Abhinav Kandala et al. “Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets”. In: *Nature* 549.7671 (Sept. 2017), pp. 242–246. ISSN: 1476-4687. DOI: 10.1038/nature23879. URL: http://dx.doi.org/10.1038/nature23879.

[116] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*. 2023. arXiv: 1604.00772 [cs.LG]. URL: https://arxiv.org/abs/1604.00772.

[117] Fedor Levkovich-Maslyuk. “The Bethe ansatz”. In: *Journal of Physics A: Mathematical and Theoretical* 49.32 (July 2016), p. 323004. ISSN: 1751-8121. DOI: 10.1088/1751-8113/49/32/323004. URL: http://dx.doi.org/10.1088/1751-8113/49/32/323004.

[118] Alejandro Sopena et al. “Algebraic Bethe Circuits”. In: *Quantum* 6 (Sept. 2022), p. 796. ISSN: 2521-327X. DOI: 10.22331/q-2022-09-08-796. URL: http://dx.doi.org/10.22331/q-2022-09-08-796.

[119] Roberto Ruiz et al. “The Bethe Ansatz as a Quantum Circuit”. In: *Quantum* 8 (May 2024), p. 1356. ISSN: 2521-327X. DOI: 10.22331/q-2024-05-23-1356. URL: http://dx.doi.org/10.22331/q-2024-05-23-1356.

[120] V. Murg, V. E. Korepin, and F. Verstraete. “Algebraic Bethe ansatz and tensor networks”. In: *Physical Review B* 86.4 (July 2012). ISSN: 1550-235X. DOI: 10.1103/physrevb.86.045125. URL: http://dx.doi.org/10.1103/PhysRevB.86.045125.

[121] Rafael I. Nepomechie. *Bethe ansatz on a quantum computer?* 2021. arXiv: 2010.01609 [quant-ph]. URL: https://arxiv.org/abs/2010.01609.

[122] Marcus Cramer et al. “Efficient quantum state tomography”. In: *Nature communications* 1.1 (2010), p. 149. DOI: 10.1038/ncomms1147(2010). URL: https://doi.org/10.1038/ncomms1147%20(2010).

[123] Oriel Kiss et al. “Statistics of topological defects across a phase transition in a superconducting quantum processor”. In: *Quantum Sci. Technol.* (June 2025), p. 035037. DOI: 10.1088/2058-9565/addf75.

[124] Miroslav Urbanek et al. “Mitigating Depolarizing Noise on Quantum Computers with Noise-Estimation Circuits”. In: *Phys. Rev. Lett.* 127 (27 Dec. 2021), p. 270502. DOI: 10.1103/PhysRevLett.127.270502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.127.270502.

[125] Oriel Kiss, Michele Grossi, and Alessandro Roggero. “Quantum error mitigation for Fourier moment computation”. In: *Phys. Rev. D* 111 (3 Feb. 2025), p. 034504. DOI: 10.1103/PhysRevD.111.034504. URL: https://link.aps.org/doi/10.1103/PhysRevD.111.034504.

[126] Z. Cai and S. Benjamin. “Constructing Smaller Pauli Twirling Sets for Arbitrary Error Channels”. In: *Sci Rep* 9 (2019), p. 1128. DOI: https://doi.org/10.1038/s41598-019-46722-7.

[127] Paul D. Nation et al. “Scalable Mitigation of Measurement Errors on Quantum Computers”. In: *PRX Quantum* 2 (4 Nov. 2021), p. 040326. DOI: 10.1103/PRXQuantum.2.040326. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.040326.

[128] Ewout van den Berg, Zlatko K. Minev, and Kristan Temme. “Model-free readout-error mitigation for quantum expectation values”. In: *Phys. Rev. A* 105 (3 Mar. 2022), p. 032620. DOI: 10.1103/PhysRevA.105.032620. URL: https://link.aps.org/doi/10.1103/PhysRevA.105.032620.

[129] Double-bracket quantum algorithms in Qibo. *qiboteam/qibo-dbqa*. 2025.

[130] Stavros Efthymiou et al. “Qibo: a framework for quantum simulation with hardware acceleration”. In: *Quantum Science and Technology* 7.1 (Dec. 2021), p. 015018. DOI: 10.1088/2058-9565/ac39f5. URL: https://doi.org/10.1088%2F2058-9565%2Fac39f5.

[131] Jeongrak Son et al. *Quantum Dynamic Programming*. 2024. arXiv: 2403.09187 [quant-ph]. URL: https://arxiv.org/abs/2403.09187.

[132] Roeland Wiersema and Nathan Killoran. “Optimizing quantum circuits with Riemannian gradient flow”. In: *Phys. Rev. A* 107 (6 June 2023), p. 062421. DOI: 10.1103/PhysRevA.107.062421. URL: https://link.aps.org/doi/10.1103/PhysRevA.107.062421.

[133] James Stokes et al. “Quantum Natural Gradient”. In: *Quantum* 4 (May 2020), p. 269. ISSN: 2521-327X. DOI: 10.22331/q-2020-05-25-269. URL: http://dx.doi.org/10.22331/q-2020-05-25-269.

[134] Qblox. https://www.qblox.com.

[135] QuantumMachines. https://www.quantum-machines.co/.

[136] ZurichInstruments. https://www.zhinst.com/others/en/quantum-computing-systems/qccs. 2023.

[137] Leandro Stefanazzi et al. “The QICK (Quantum Instrumentation Control Kit): Readout and control for qubits and detectors”. In: *Review of Scientific Instruments* 93.4 (Apr. 2022). DOI: 10.1063/5.0076249. URL: https://doi.org/10.1063/5.0076249.

[138] Rodolfo Carobene et al. *qiboteam/qibosoq: Qibosoq 0.0.3*. Version v0.0.3. July 2023. DOI: 10.5281/zenodo.8126172. URL: https://doi.org/10.5281/zenodo.8126172.

[139] Qblox. https://qblox-qblox-instruments.readthedocs-hosted.com/en/master/getting_started/product_overview.html#cluster.

[140] Qblox. https://qblox-qblox-instruments.readthedocs-hosted.com/en/master/cluster/qrm_rf.html. 2023.

[141] Qblox. https://qblox-qblox-instruments.readthedocs-hosted.com/en/master/cluster/qcm_rf.html. 2023.

[142] Qblox. https://qblox-qblox-instruments.readthedocs-hosted.com/en/master/cluster/qcm.html. 2023.

[143] Qblox. https://qblox-qblox-instruments.readthedocs-hosted.com/en/master/cluster/synchronization.html#synq.

[144] Qcodes. https://qcodes.github.io/Qcodes/. 2023.

[145] Qblox. https://qblox-qblox-instruments.readthedocs-hosted.com/en/master/tutorials/q1asm_tutorials.html. 2023.

[146] OPX+. https://www.quantum-machines.co/products/opx/.

[147] ZurichInstruments. https://www.zhinst.com/others/en/products/shfqc-qubit-controller. 2023.

[148] Johannes Herrmann et al. *Frequency Up-Conversion Schemes for Controlling Superconducting Qubits*. 2022. DOI: 10.48550/arXiv.2210.02513. URL: https://doi.org/10.48550/arXiv.2210.02513.

[149] ZurichInstruments. https://www.zhinst.com/others/en/products/hdawg-arbitrary-waveform-generator. 2023.

[150] ZurichInstruments. https://www.zhinst.com/others/en/products/pqsc-programmable-quantum-system-controller. 2023.

[151] Xilinx-(AMD). *RFSoC 4x2 specifications*. https://www.xilinx.com/support/university/xup-boards/RFSoC4x2.html. 2022.

[152] Xilinx-(AMD). *ZCU111 specifications*. https://www.xilinx.com/products/boards-and-kits/zcu111.html. 2022.

[153] Xilinx-(AMD). *ZCU216 specifications*. https://www.xilinx.com/products/boards-and-kits/zcu216.html. 2022.

[154] P S V Naidu. *Modern Digital Signal Processing*. Alpha Science International, Sept. 2003.

[155] Mahdi Naghiloo. *Introduction to Experimental Quantum Measurement with Superconducting Qubits*. 2019. DOI: 10.48550/arXiv.1904.09291. URL: https://doi.org/10.48550/arXiv.1904.09291.

[156] Frank Arute et al. “Quantum Supremacy using a Programmable Superconducting Processor”. In: *Nature* 574 (2019), pp. 505–510. DOI: 10.1038/s41586-019-1666-5. URL: https://doi.org/10.1038/s41586-019-1666-5.

[157] PV Klimov et al. “Fluctuations of energy-relaxation times in superconducting qubits”. In: *Physical review letters* 121.9 (2018), p. 090502.

[158] Cora N. Barrett et al. “Learning-Based Calibration of Flux Crosstalk in Transmon Qubit Arrays”. In: *Physical Review Applied* 20.2 (Aug. 2023). ISSN: 2331-7019. DOI: 10.1103/physrevapplied.20.024070. URL: http://dx.doi.org/10.1103/PhysRevApplied.20.024070.

[159] A. A. Houck et al. “Controlling the Spontaneous Emission of a Superconducting Transmon Qubit”. In: *Phys. Rev. Lett.* 101 (8 Aug. 2008), p. 080502. DOI: 10.1103/PhysRevLett.101.080502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.101.080502.

[160] Koch, Christiane P. et al. “Quantum optimal control in quantum technologies. Strategic report on current status, visions and goals for research in Europe”. In: *EPJ Quantum Technol.* 9.1 (2022), p. 19. DOI: 10.1140/epjqt/s40507-022-00138-x. URL: https://doi.org/10.1140/epjqt/s40507-022-00138-x.

[161] J. Helsen et al. “General Framework for Randomized Benchmarking”. In: *PRX Quantum* 3 (2 June 2022), p. 020357. DOI: 10.1103/PRXQuantum.3.020357. URL: https://link.aps.org/doi/10.1103/PRXQuantum.3.020357.

[162] D. J. Egger and F. K. Wilhelm. “Adaptive Hybrid Optimal Quantum Control for Imprecisely Characterized Systems”. In: *Phys. Rev. Lett.* 112 (24 June 2014), p. 240503. DOI: 10.1103/PhysRevLett.112.240503. URL: https://link.aps.org/doi/10.1103/PhysRevLett.112.240503.

[163] J. Kelly et al. “Optimal Quantum Control Using Randomized Benchmarking”. In: *Phys. Rev. Lett.* 112 (24 June 2014), p. 240504. DOI: 10.1103/PhysRevLett.112.240504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.112.240504.

[164] PostgreSQL Global Development Group. https://www.postgresql.org/.

[165] E. Knill et al. “Randomized benchmarking of quantum gates”. In: *Phys. Rev. A* 77 (1 Jan. 2008), p. 012307. DOI: 10.1103/PhysRevA.77.012307. URL: https://link.aps.org/doi/10.1103/PhysRevA.77.012307.

[166] Michael A Nielsen. “A simple formula for the average gate fidelity of a quantum dynamical operation”. In: *Physics Letters A* 303.4 (2002), pp. 249–252. ISSN: 0375-9601. DOI: https://doi.org/10.1016/S0375-9601(02)01272-0. URL: https://www.sciencedirect.com/science/article/pii/S0375960102012720.

[167] S. Kehrein. *The flow equation approach to many-particle systems*. Vol. 217. 2006. DOI: 10.1007/3-540-34068-8.

[168] R.W. Brockett. “Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems”. In: *Linear Algebra and its Applications* 146 (1991), pp. 79–91. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(91)90021-N. URL: https://www.sciencedirect.com/science/article/pii/002437959190021N.

[169] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th A nniversary Edition*. Cambridge University Press, 2010.

[170] Stavros Efthymiou et al. “Qibolab: an open-source hybrid quantum operating system”. In: *Quantum* 8 (Feb. 2024), p. 1247. ISSN: 2521-327X. DOI: 10.22331/q-2024-02-12-1247. URL: http://dx.doi.org/10.22331/q-2024-02-12-1247.

[171] Andrea Pasquale et al. *Beyond full statevector simulation with Qibo*. 2024. arXiv: 2408.00384 [quant-ph]. URL: https://arxiv.org/abs/2408.00384.

[172] Edoardo Pedicillo et al. *An open-source framework for quantum hardware control*. 2024. arXiv: 2407.21737 [quant-ph]. URL: https://arxiv.org/abs/2407.21737.

[173] Andrea Pasquale et al. *Towards an open-source framework to perform quantum calibration and characterization*. 2024. arXiv: 2303.10397 [quant-ph]. URL: https://arxiv.org/abs/2303.10397.

[174] Adriá n Pérez-Salinas et al. “Determining the proton content with a quantum computer”. In: *Physical Review D* 103.3 (Feb. 2021). DOI: 10.1103/physrevd.103.034027. URL: https://doi.org/10.1103%2Fphysrevd.103.034027.

[175] Carlos Bravo-Prieto et al. “Style-based quantum generative adversarial networks for Monte Carlo events”. In: *Quantum* 6 (Aug. 2022), p. 777. ISSN: 2521-327X. DOI: 10.22331/q-2022-08-17-777. URL: http://dx.doi.org/10.22331/q-2022-08-17-777.

[176] Matteo Robbiati, Juan M. Cruz-Martinez, and Stefano Carrazza. *Determining probability density functions with adiabatic quantum computing*. 2023. DOI: 10.48550/arXiv.2303.11346. URL: https://doi.org/10.48550/arXiv.2303.11346.

[177] Juan M Cruz-Martinez, Matteo Robbiati, and Stefano Carrazza. “Multi-variable integration with a variational quantum circuit”. In: *Quantum Science and Technology* 9.3 (June 2024), p. 035053. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ad5866. URL: http://dx.doi.org/10.1088/2058-9565/ad5866.

[178] Simone Bordoni et al. “Long-Lived Particles Anomaly Detection with Parametrized Quantum Circuits”. In: *Particles* 6.1 (Feb. 2023), pp. 297–311. ISSN: 2571-712X. DOI: 10.3390/particles6010016. URL: http://dx.doi.org/10.3390/particles6010016.

[179] Vasilis Belis et al. “Quantum anomaly detection in the latent space of proton collision events at the LHC”. In: *Communications Physics* 7.1 (2024), p. 334. DOI: 10.1038/s42005-024-01811-6. URL: https://doi.org/10.1038/s42005-024-01811-6.

[180] Jorge J. Martinez de Lejarza et al. “Loop Feynman integration on a quantum computer”. In: *Phys. Rev. D* 110 (7 Oct. 2024), p. 074031. DOI: 10.1103/PhysRevD.110.074031. URL: https://link.aps.org/doi/10.1103/PhysRevD.110.074031.

[181] Matteo Robbiati et al. *A quantum analytical Adam descent through parameter shift rule using Qibo*. 2022. arXiv: 2210.10787 [quant-ph]. URL: https://arxiv.org/abs/2210.10787.

[182] Matteo Robbiati et al. *Real-time error mitigation for variational optimization on quantum hardware*. 2023. arXiv: 2311.05680 [quant-ph]. URL: https://arxiv.org/abs/2311.05680.

[183] Alessandro D’Elia et al. “Characterization of a Transmon Qubit in a 3D Cavity for Quantum Machine Learning and Photon Counting”. In: *Applied Sciences* 14.4 (2024). ISSN: 2076-3417. DOI: 10.3390/app14041478. URL: https://www.mdpi.com/2076-3417/14/4/1478.

[184] François Chollet et al. *Keras*. https://keras.io. 2015.

[185] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[186] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. DOI: 10.48550/arXiv.1412.6980. URL: https://doi.org/10.48550/arXiv.1412.6980.

[187] Maria Schuld et al. “Evaluating analytic gradients on quantum hardware”. In: *Physical Review A* 99.3 (Mar. 2019). DOI: 10.1103/physreva.99.032331. URL: https://doi.org/10.1103%2Fphysreva.99.032331.

[188] K. Mitarai et al. “Quantum circuit learning”. In: *Physical Review A* 98.3 (Sept. 2018). DOI: 10.1103/physreva.98.032309. URL: https://doi.org/10.1103%2Fphysreva.98.032309.

[189] Farrokh Vatan and Colin Williams. “Optimal quantum circuits for general two-qubit gates”. In: *Phys. Rev. A* 69 (3 Mar. 2004), p. 032315. DOI: 10.1103/PhysRevA.69.032315. URL: https://link.aps.org/doi/10.1103/PhysRevA.69.032315.

[190] Vivek V Shende, Stephen S Bullock, and Igor L Markov. “Synthesis of quantum logic circuits”. In: *Proceedings of the 2005 Asia and South Pacific D esign Automation Conference*. 2005, pp. 272–275.

[191] Jorge V. Jose. “40 years of Berezinskii-Kosterlitz-Thouless Theory”. In: 2013. URL: https://api.semanticscholar.org/CorpusID:117419125.

[192] Steven R. White and Ian Affleck. “Dimerization and incommensurate spiral spin correlations in the zigzag spin chain: Analogies to the Kondo lattice”. In: *Phys. Rev. B* 54 (14 Oct. 1996), pp. 9862–9869. DOI: 10.1103/PhysRevB.54.9862. URL: https://link.aps.org/doi/10.1103/PhysRevB.54.9862.

[193] Hanna Westerheim et al. “Dual-VQE: A quantum algorithm to lower bound the ground-state energy”. In: *arXiv preprint arXiv:2312.03083* (2023). DOI: https://doi.org/10.48550/arXiv.2312.03083. URL: https://arxiv.org/abs/2312.03083.

[194] Christian Kokail et al. “Self-verifying variational quantum simulation of lattice models”. In: *Nature* 569.7756 (2019), pp. 355–360. DOI: 10.1038/s41586-019-1177-4. URL: https://www.nature.com/articles/s41586-019-1177-4.

[195] Kentaro Yamamoto et al. “Demonstrating Bayesian quantum phase estimation with quantum error detection”. In: *Phys. Rev. Res.* 6 (1 Feb. 2024), p. 013221. DOI: 10.1103/PhysRevResearch.6.013221. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.6.013221.

[196] Kentaro Yamamoto. *Private communication*. 2024.